Hidden in Plain Sight: Automatically Identifying Security Requirements from Natural Language Artifacts

Maria Riaz, Jason King, John Slankas, and Laurie Williams
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
[mriaz, jtking, john.slankas, laurie_williams]@ncsu.edu

***Abstract*—Natural language artifacts, such as requirements specifications, often explicitly state the security requirements for software systems. However, these artifacts may also imply additional security requirements that developers may overlook but should consider to strengthen the overall security of the system. *The goal of this research is to aid requirements engineers in producing a more comprehensive and classified set of security requirements by (1) automatically identifying security-relevant sentences in natural language requirements artifacts, and (2) providing context-specific security requirements templates to help translate the security-relevant sentences into functional security requirements.* Using machine learning techniques, we have developed a tool-assisted process that takes as input a set of natural language artifacts. Our process automatically identifies security-relevant sentences in the artifacts and classifies them according to the security objectives, either explicitly stated or implied by the sentences. We classified 10,963 sentences in six different documents from healthcare domain and extracted corresponding security objectives. Our manual analysis showed that 46% of the sentences were security-relevant. Of these, 28% explicitly mention security while 72% of the sentences are functional requirements with security implications. Using our tool, we correctly predict and classify 82% of the security objectives for all the sentences (precision). We identify 79% of all security objectives implied by the sentences within the documents (recall). Based on our analysis, we develop context-specific templates that can be instantiated into a set of functional security requirements by filling in key information from security-relevant sentences.**

***Index Terms*— Security, requirements, objectives, templates, access control, auditing, text classification, constraints, natural language parsing.**

I. Introduction

Security requirements provide a foundation for building secure software systems. Despite the availability of methods and processes for security requirements engineering [1], teams often do not focus on security during early stages of software development [2]. Existing approaches outline the various steps involved in identifying security requirements, but leave the task of executing these steps for the requirements engineer, who may not be an expert in security. Many security requirements are functional in nature and need to be incorporated into system design to ensure data and system security. Natural language requirements artifacts, such as requirements documents, often explicitly state the security requirements for software systems. However, additional sentences in these documents may have security implications [3] leading to additional security requirements.

Our research on identifying applicable security requirements for software systems is guided by the following primary motivations: Software systems that share security objectives, such as confidentiality or integrity, also have similar sets of security requirements [4]. These security requirements, if specified at the right level of abstraction, can be reusable across multiple systems, even as a set to meet the same security objective [5, 6]. Patterns and similarities in grammar or phrasing of security requirements may exist and allow the security requirements to be reused across multiple software systems with minor tweaks to content, such as different actions taken or different resources being acted upon. By first identifying both the explicit and implied security objectives of a software system, we intend to discover an abstract set of security requirements that may be considered when developing any software system that shares similar security objectives. Natural language requirements artifacts often contain security-relevant sentences that are indicative of the security objectives and security requirements of the system [3].

The goal of this research is to aid requirements engineers in producing a more comprehensive and classified set of security requirements by (1) automatically identifying security-relevant sentences in natural language requirements artifacts, and (2) providing context-specific security requirements templates to help translate the security-relevant sentences into functional security requirements.

We present a tool-assisted process, Security Discoverer (SD), that incorporates machine learning techniques to identify a set of security requirements for an input set of natural language requirements artifacts. We classify the sentences in the input in terms of their security objectives (such as, confidentiality, integrity, availability). This classification can be used to guide the analyst in creating an appropriate set of security requirements and in organizing the resultant set of security requirements. Using this approach, we have classified 10,963 sentences in six natural language artifacts from the electronic healthcare domain in terms of their corresponding

978-1-4799-3032-6/14/$31.00 © 2014 IEEE

RE 2014, Karlskrona, Sweden

security objectives. By observing similarities and abstracting common elements in the classified set of security-relevant sentences, we empirically derive a set of context-specific security requirements templates. Our tool suggests applicable templates for instantiation by the requirements engineers to generate the security requirements.

For example, consider the sentence "The system shall provide a means to edit discharge instructions for a particular patient."[1] This sentence does not explicitly state a security requirement but implies security requirements for confidentiality (*of patient's discharge instructions*), integrity (*when editing*) and accountability (*who performed the edits*). Security requirements that can be generated by instantiating corresponding templates include:

- "The system shall enforce access privileges that enable authorized users to edit discharge instructions for a particular patient." (confidentiality)
- "The system shall log every time discharge instructions for a particular patient are edited." (accountability)

We use the following research questions to guide us in meeting our research goal:

RQ1: What are the core categories of security objectives in existing literature that should be considered during the security requirements engineering process?

RQ2: How often are security objectives explicitly stated or implied in natural language requirements artifacts?

RQ3: How effectively can security objectives be identified and extracted from natural language project documents?

RQ4: What similarities (words, phrases, grammatical structure, etc.) exist among security-relevant sentences for each security objective?

RQ5: What common templates for specifying functional security requirements can be empirically derived from the security-relevant sentences?

Our research contributes the following:

- A set of core categories of security objectives that requirements engineers should consider during the requirements engineering process.
- A tool-assisted process to aid requirements engineers in identifying and classifying security relevant-sentences in terms of security objectives.
- A set of context-specific security requirements templates to help requirements engineers translate classified security-relevant sentences into functional security requirements that meet specific objectives.

The rest of this paper is organized as follows: Section II reviews the background and related work. Section III presents the security objectives to address RQ1 followed by our proposed tool-assisted process, Security Discoverer, in Section IV. In Section V, we describe our research methodology. Section VI presents results and evaluation to address RQ2-RQ4. We present the set of context-specific templates based on our analysis in Section VII to address RQ5. Section VIII discusses threats to validity for our study. Finally, Section IX concludes the paper in addition to outlining future directions.

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

II. Background and Related Work

In this section, we discuss the background of security objectives, and related work in requirement classifications and security requirements engineering.

A. Security Objectives and Requirements

Security objectives are the security goals or desired security properties of a system [7]. Security requirements are functional and non-functional requirements that operationalize security objectives without specifying how to achieve those objectives. Functional security requirements describe the desired security behavior of a system [8] and, if incorporated, can achieve the corresponding security objectives. For this paper, we use the term security requirements to mean functional security requirements.

Firesmith [9] argues security requirements can be reusable across multiple systems and has proposed the use of parameterized templates to model reusable security requirements. Mellado et al., [6] argue the effectiveness of reusing related security requirements that act as a group to meet security objectives. In our work, we group security-relevant sentences in terms of security objectives and provide context-specific templates to meet these objectives, building on the observations from previous work.

B. Identifying Security Requirements

Security Requirements Engineering (SRE) has gained focus in recent years with emphasis on identifying security requirements early on in the software development lifecycle. Mellado et al. have conducted a systematic review of SRE approaches [10] to summarize existing methodologies. Fabian et al. also provide a comparison of SRE methods [1]. These methods include lifecycle-based approaches such as Microsoft SDL [11] as well as methods that solely focus on SRE, such as the SQUARE method [12], which provides a framework for generating non-functional security requirements. Other approaches for identifying security requirements include misuse or abuse cases [13], anti-goals [14], and assurance arguments [15].

A recent analysis of various SRE methods [16] indicates that despite the availability of a number of SRE methods, only a handful of these methods have been used in practice. Their findings, based on feedback from practitioners across various organizations, also highlight that SRE as a tool-assisted process may facilitate secure software development. Efforts to automate parts of SRE process have resulted in organizational learning approach to SRE [17] that identifies when a security requirement is added to a natural language artifact. This helps build a repository of security requirements for consideration and reuse in subsequent projects. Our process goes beyond identifying explicit security requirements in text. We identify implied security-relevant sentences in natural language artifacts and associated security objectives. We also provide context-specific templates to identify functional security requirements to meet the identified objectives.

Another focus area related to security requirements engineering has been on extracting security requirements from regulatory texts ([18, 19]). Other researchers have explored using natural language to generate access control policies ([20, 21]). The focus of 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.

C. Requirements 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 [22], non-functional requirement (NFR) classification first appeared in the literature in 2006 [23]. 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, successfully identifying 81% of the NFRs in the dataset. However, their precision was 0.1244 indicating a large number of false positives. Other researchers [24, 25] built upon this work by using the same dataset as Cleland-Huang, but adopt naïve Bayes and Support Vector Machine (SVM) classifiers. Both experiments reported higher scores for precision than the original research. We evaluated a number of different machine learning algorithms for text classification and decided upon a k-NN classifier as its performance matched that of an SVM classifier and analysts can more easily understand where results were derived in the k-NN classifier.

III. 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 intent of the sentence as well as possible requirements and mechanisms to establish that intent. Security objectives of software systems 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. While certain sets of security objectives are widely known such as "Confidentiality, Integrity, and Availability (CIA) Triad", we want to ensure the completeness of our security objective set.

RQ1: What are the core categories of security objectives in existing literature that should be considered during the security requirements engineering process?

We examined six security standards ([8, 26-30]), two taxonomies of security objectives and requirements ([9, 14]) and two security seminal papers and books ([7, 31]). We created a list of security objectives identified from the above sources. 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 sentences from the set of documents we analyzed (see Section V.A) that indicate the presence of corresponding objective. The examples are numbered as: <Document ID>-<Security Objective Abbreviation>.<#>.

Confidentiality (C) ([7-9, 14, 26, 29]): The degree to which the "data is disclosed only as intended" [7]

Example: "The system should provide the ability to electronically capture patient data including medications, vital signs, and other data as structured data" (EA-C.1)

Integrity (I) ([7-9, 14, 26, 29]): "The degree to which a system or component guards against improper modification or destruction of computer programs or data." [28]

Example: "…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." (ED-I.1)

Identification & Authentication (IA) ([8, 9, 14, 26, 29]): The need to establish that "a claimed identity is valid" for a user, process or device. [27]

Example: "The system shall authenticate the user before any access to Protected Resources (e.g. PHI) is allowed, including when not connected to a network e.g. mobile devices." (CT-IA.2)

Availability (A) ([7-9, 14, 26, 29]): "The degree to which a system or component is operational and accessible when required for use." [32]

Example: "Provides business continuity in the situation where the EHR system is not available by providing access to the last available clinically relevant patient data in the EHR." (NU-A.3)

Accountability (AY) ([7-9, 14, 26, 29]): Degree to which actions affecting software assets "can be traced to the actor responsible for the action" [7]

Example: "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." (ED-AY.1)

Privacy (PR) ([8, 9, 14]): The degree to which an actor can understand and control how their information is used.

Example: "Nurses need to provide legitimate care in crisis situations that may go against prior patient consent directives ("break the glass" situations)". (NU-PR.2)

IV. Security Discoverer

We now present our process, Security Discoverer (SD), and its associated tool.

A. Overview

We have developed a four-step tool-assisted process for identifying security requirements. As a first step, our tool takes natural language requirements artifacts (requirement specifications, feature requests, etc.) and a trained classifier for the current problem domain as input. The tool parses the artifacts as text sentences and identifies which (if any) security objectives relate to each sentence. The tool then presents the user with a list of applicable security requirements templates for the identified objectives. The user then selects the

appropriate templates and completes the template with details from the initial requirement. Our tool[2] provides an authoring mechanism to finalize the identified security requirements. The tool also supports traceability of generated security requirements to source sentences in input artifacts. We explain the steps of the SD process below.

B. Step 1: Pre-process Artifacts

We import a natural language requirement artifact into the SD tool to prepare each sentence to be classified with relevant security objective(s). A user first needs to convert the artifact into a text only format. Generally, this conversion can be accomplished through the "Save As" format within Microsoft Word or other document applications. As such, this process will not convert tables or images properly and the user will need to manually perform the conversion for those sections. Once the artifact has been prepared, the tool will first read the entire text into the system. Next, to provide additional context and features for the classifier, the tool applies a concise document grammar (Figure 1) to label each sentence in the text to a specific type:

- *title*: Sentences that follow capitalization rules for titles.
- *list start*: These sentences represent the header or description of a list that follows.
- *list element*: These sentences represent individual items contained within an ordered or unordered list. These sentences 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 sentences are not considered as titles, list starts, or list elements.

Further, we identify heading and list identifiers (e.g., "4.1.1" and "•") and remove those identifiers from sentences used in classification. These identifiers create superficial differences among sentences and can possibly skew classification results. When sentences are classified in the next step, we combine "list start" with each identified "list element" as part of the same list to provide additional context.

Within Figure 1, 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.

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

Fig. 1. Document Grammar

While we do not expect the documents to be well-formed, our process works better with shorter, well-formed sentences.

C. Step 2: Classify for Security Objectives

We classify each sentence in the input into zero or more security objectives. The classifier can be created in one of three ways: 1) training 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.

We utilize a k-NN classifier for this step. Such classifiers work by taking a majority vote of the existing classifications of the k nearest neighbors to the item under test. To determine the closest sentence(s), we apply a custom distance function based upon a modified version of Levenshtein distance [33]. Rather than using the resulting number of edits to transform one string into another as the value as the Levenshtein distance does, our metric computes the number of word transformations to change one sentence into another. Repetition of words and phrases in different sentences can help the classification process. While other machine learning algorithms can provide similar performance to k-NN classifier, the k-NN classifier provides for easier interpretation of results by requirements engineers as they can see similar sentences and associated classifications. The distance metric can also be used within distance-based clustering algorithms for further analysis.

Once the classification is complete, the user may review the predicted security objectives for the security-relevant sentences. If necessary, the user can correct the classified objectives within the tool.

D. Step 3: Select Context-specific Templates

Once the security objectives have been identified for a given sentence, the tool presents the user with a list of context-specific security requirements templates for the security objectives and values (such as action or time) present within the sentence. The security templates are further discussed in Section VII. A sample template for the objective "accountability" is displayed in Table VI. The user selects which templates apply to the given sentence. 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.

E. Step 4: Generate Requirement Sentences

Once the requirement templates have been selected by the user, the tool presents the requirements 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-related sentence specifies "during business hours", tool detects the time period and would automatically place that phrase into the generated requirement template. The tool maps generated requirements to source sentences to produce a traceability matrix.

[2] Source code available at: http://go.ncsu.edu/securitydiscoverer/

V. Research Methodology

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

A. Study Documents

We have selected six freely-available natural language requirements artifacts from the electronic healthcare domain, listed in Table I. Due to regulation and standardization, documents in healthcare domain generally tend to be well-formed. However, we select a variety of document types, including feature requests that are not well-formed. The selected documents are from USA and Canada. Use of different spellings for same words (e.g., color vs. colour) may also affect the performance of our classifier.

B. Study Oracle

We have developed a study oracle to train our tool and evaluate the performance of our classifier. To create the study oracle, three researchers manually read each natural language sentence in the six healthcare documents, and classify the sentence 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 sentences using natural language processing (see Section IV.B).
3) Manually classify each sentence in a document.
   a) *Classification Phase*: For each document, two researchers individually classify each sentence to identify security objectives that apply to the sentence. The classification phase results in the creation of two separate output files (one per researcher) for each input document.
   b) *Validation Phase*: A third researcher generates a difference report from the classifications of the other two researchers. This third researcher resolves the differences by communicating with the original two researchers to generate consensus for creating a final, consolidated classified corpus document.

Table I provides a document-wise breakdown of sentences classified per security objective. Each sentence could be classified in terms of zero or more security objectives. The researchers spent a total of approximately 160 person-hours to create and validate the oracle that we use for further analysis. Researchers had a moderate agreement [34] on whether a sentence was security-relevant or not (indicated by a kappa score of 0.54). Of the security-relevant sentences, we had an almost perfect agreement in terms of whether a sentence explicitly talks about security or implies a need for security (kappa score of 0.85). We also had a fair agreement on classification of objectives for each sentence (kappa score of 0.32; kappa score tends to decrease as classification categories increase). Requirements engineers looking to adopt our process can incrementally build upon our existing classifier or train a shared classifier for their domain over time by classifying security-relevant sentences in natural language artifacts. Performance of the classifier is expected to improve as the number of classified sentences increases. Incrementally evolving the classifier as a community will save time and effort upfront while creating a knowledgebase of security-relevant sentences and security objectives of software systems.

C. Study Procedure

Once the study oracle has been created, we execute a variety of classifiers (our k-NN classifier and from Weka [35] - a multinomial naïve Bayes classifier, and a SMO - sequential 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. To compute these values, we first need to categorize the classifier's predictions into three categories. 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. 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: $R = TP/(TP + FN)$. F_1 measure is the harmonic mean of precision and recall, giving equal weight to both: $F_1 = 2 \times \frac{P \times R}{P + R}$.

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 [36] 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. We directly utilized Weka classifiers through the available Java APIs utilizing their default options. 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 [37] attribute selector within Weka. Yang and Pedersen [38] found information gain to be the most effective method for feature selection in text classification.

D. Security Requirements Template Extraction

We analyze the classified set of sentences associated with each security objective and identify commonalities in those sentences based on the following attributes:

- Common patterns and themes in sentence structure
- Keywords in the sentences
- Clustering of sentences (k-mediods/LDA)

Based on our analysis, we develop templates that would allow incorporating security requirements to meet corresponding security objectives while maintaining neutrality to the mechanisms. We discuss the templates in Section VII.

TABLE I. DOCUMENTS AND ASSOCIATED SECURITY OBJECTIVE COUNTS

Doc. ID	Document Title	#Sentences	Security Objectives						
			C	I	IA	A	AY	PR	None
CT	Certification Commission for Healthcare Information Technology (CCHIT) Certified 2011 Ambulatory EHR Criteria[3]	331	252	214	19	14	260	5	6
ED	Emergency Department Information Systems Functional Document[4]	2328	1162	1173	75	35	1354	76	773
NU	Pan-Canadian Nursing EHR Business and Functional Elements Supporting Clinical Practice[5]	264	67	77	4	26	43	10	96
OR	Open Source Clinical Application Resource (OSCAR) Feature Requests[6]	5081	696	974	104	10	1184	18	3735
PS	Canada Health Infoway Electronic Health Record (EHR) Privacy and Security Requirements[7]	1623	146	120	43	31	149	85	928
VL	Virtual Lifetime Electronic Record User Stories[8]	1336	693	731	13	19	797	10	375
	Total # (%):	10963	3016 (27%)	3289 (30%)	258 (~2%)	135 (~1%)	3787 (34%)	204 (2%)	5913 (54%)

VI. EVALUATION

In this section, we address research questions RQ2 - RQ4.

RQ2: How often are security objectives explicitly stated or implied in natural language requirements artifacts?

Based on the study oracle, we identified that 46% of the sentences in input artifacts relate to security. Given that we selected documents from industry standards and best practices related to the healthcare domain (which involves protected health information), security-relevant sentences intuitively form a large proportion of the document. Of all the security-relevant sentences, only 28% *explicitly* mention security (13% of total sentences, similar to our earlier findings [3]), while 72% are functional requirements with security implication (an additional 33% of total sentences). If implied security objectives are not considered, requirements engineers may overlook key security requirements. Table II provides a document-wise breakdown of sentences and whether security objectives were implied or explicitly stated.

From the security relevant sentences, we identified the security objectives that are implied by each sentence. The top three implied security objectives are accountability (34% of all sentences), integrity (30%) and confidentiality (27%). Privacy (2%), identification & authentication (~2%), and availability (~1%) objectives were implied by only a small percentage of all sentences. Our results indicate that 93% of the security-relevant sentences implied more than one security objective. Table III presents the 10 most frequently occurring security objective groups. Confidentiality and accountability each appear in 7 of 10 top objective groups, suggesting that confidentiality and accountability are common security objectives for healthcare systems. Integrity appears in 6 of 10 top objective groupings.

[3] https://www.cchit.org/
[4] http://www.hl7.org/
[5] https://www.infoway-inforoute.ca/
[6] http://oscarcanada.org/
[7] https://www.infoway-inforoute.ca/
[8] http://www.va.gov/vler/

The confidentiality, integrity, and accountability objectives appear *together* in the classifications of 2,232 sentences (20% of all sentences classified), suggesting a strong relationship among the three. For example, the sentence "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.

TABLE II. IMPLICIT AND EXPLICIT SECURITY-RELEVANT SENTENCES

Doc ID	Total Sente-nces	Explicit Security # (%)	Implicit Security # (%)	Total Security #(%)	Not Security Related
CT	331	89 (27%)	236 (71%)	325 (98%)	6 (2%)
ED	2328	274 (12%)	1281 (55%)	1555 (67%)	773 (33%)
NU	264	41 (16%)	127 (48%)	264 (64%)	96 (36%)
OR	5081	174 (3%)	1172 (23%)	1346 (26%)	3735 (74%)
PS	1623	628 (39%)	67 (4%)	695 (43%)	928 (57%)
VL	1336	185 (14%)	776 (58%)	961 (72%)	375 (28%)
Tot-al	**10963**	**1391 (13%)**	**3659 (33%)**	**5050 (46%)**	**5913 (54%)**

Confidentiality and accountability appear together in the classifications of 2,859 sentences (26% of all sentences classified). 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 study oracle, sentences that involve create/read/update/delete actions upon sensitive data are often classified as implying both confidentiality and accountability.

Integrity and accountability appear together for 3,119 sentences (28.5% of all sentences classified). With respect to 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.

TABLE III. FREQUENTLY OCCURRING OBJECTIVE GROUPS

Frequency # (% sec-relevant)	Objective Group
2232 (44%)	Confidentiality, Integrity, Accountability
702 (14%)	Integrity, Accountability
443 (9%)	Confidentiality, Accountability
106 (2%)	Confidentiality, Integrity
104 (2%)	Confidentiality, Identification & Authentication
98 (2%)	Confidentiality, Accountability, Privacy
95 (~2%)	Integrity, Accountability, Privacy
90 (~2%)	Integrity, Identification & Authentication, Accountability
86 (~2%)	Confidentiality, Identification & Authentication, Accountability
83 (~2%)	Confidentiality, Integrity, Privacy

RQ3: How effectively can security objectives be identified and extracted from selected set of documents?

We use recall and precision as measures to assess effectiveness. Table IV presents the results of running the four classifiers against the six documents using a ten-fold cross validation. Creating a "Combined" ensemble classifier demonstrated a slight performance gain over just using the Weka SMO classifier. The "Combined" classifier uses the results of the *k*-NN classifier if relatively close sentences were found. Otherwise, the "Combined" classifier uses a majority vote of the three classifiers. 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 all the security objectives associated with the sentences it classified. The recall score of .79 means that it found 79% of all of the possible objectives. From an error perspective, the precision score implies that 18% of the identified objectives an analyst examines would be false positives, and 21% of the possible objectives were not found.

RQ4: What similarities (words, phrases, grammatical structure, etc.) exist among security-relevant sentences for each security objective??

Overall, keywords are the primary indicator of security objectives for identification/authentication, availability, and privacy. However, for many confidentiality, integrity, and accountability sentences, the grammatical structure of the sentence is often the same. We use these similarities in grammatical structure and keywords within the sentences of each security objective to develop a set of context-specific templates for composing security requirements. We discuss the proposed templates in section VII.

Table V 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.

TABLE IV. 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	.82	.79	.80

Keywords "system", "provide", and "ability" commonly appear in sentences classified as confidentiality, integrity, and/or accountability. Sentences 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 sentence involves access to medications (protected health information), the sentence is classified as implying a confidentiality objective. Likewise, since the sentence involves interacting with protected information, the integrity of the data must be maintained. Finally, since the sentence 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 sentences for confidentiality, integrity, and accountability share a common grammatical pattern, sentences 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. 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 classified sentences for this objective. Instead, common keywords that suggest privacy objective 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. Disclosure of protected information suggests that a user has consented to disclose given information to a third-party.

TABLE V. TOP 20 KEYWORDS BY SECURITY OBJECTIVES

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

VII. CONTEXT-SPECIFIC TEMPLATES

We have developed a set of context-specific templates to translate individual security objectives for each sentence into a set of concrete functional security requirements. We maintain traceability between the original sentence in the natural language artifact and generated security requirements.

RQ5: What common templates for specifying functional security requirements can be empirically derived from the security-relevant sentences?

We have extracted 19 context-specific templates[9] based on our analysis. Each template is associated with a particular security objective and identifies the conditions under which the template becomes applicable (e.g., based on the subject, action or resource in the security-relevant sentence). Each template also provides one or more reusable parameterized security requirements that can be filled-in to generate system-specific functional security requirements. The context-specific templates, grouped by security objectives, are named below. Details of the templates are available online. [9]

- *Confidentiality*: *C1*-authorized access; *C2*-during storage; *C3*-during transmission;
- *Integrity*: *I1*- read-type actions; *I2*- write-type actions; *I3*-delete actions; *I4*-unchangeable resources;
- *Availability*: *A1*-availability of data; *A2*-appropriate response time; *A3*-service availability; *A4*-backup and recovery capabilities; *A5*-capacity and performance;
- *Identification & Authentication*: *IA1*-select context for roles; *IA2*-unique accounts; *IA3*-Authentication;
- *Accountability*: *AY1*-log transactions with sensitive data; *AY2*-log authentication events; *AY3*-log system events;
- *Privacy*: *PR1*-usage of personal information;

We list example context-specific templates, along with generated security requirements, in Table VI. Requirements analysts should consider our set of context-specific templates to determine which templates apply to each security-relevant sentence in the project documentation. We intend a requirements analyst to first identify security objectives using the SD tool on the given project artifacts before considering the templates. The tool produces a set of security objective annotations for each sentence in the documentation and suggests relevant templates. However if the security objectives are already known, the templates can be used independent of the tool as well. For example, for a sentence that the tool annotates as having an accountability objective (or objective is known a priori), the requirements analyst should consider context-specific templates for accountability (AY1, AY2 or AY3[9]). If the sentence contains a subject acting upon sensitive information, the requirements analyst should compose a total of two security requirements to fulfill the sentence's accountability objective (see Table VI).

However, the newly composed security requirements also contain related security objectives themselves. Consider the generated security requirements for AY1 in Table VI. These requirements suggest an integrity objective to prevent modification of log files (I4). The template for AY1 captures this relationship between accountability and integrity allowing the requirements analyst to consider integrity when identifying the security requirements for accountability. In Section VI, we discussed how security objectives for confidentiality, integrity, and accountability often appeared together in the classifications for over 20% of the sentences. The cross-references in our context-specific templates for composing security requirements also reflect the strong relationships among confidentiality, integrity, and accountability.

For a preliminary evaluation, we selected an example use-case from iTrust electronic health record system[10] and applied our process to generate security requirements based on the sentences in the use-case. We identified 32 additional security requirements based on the analysis of just one of the 60 documented use-cases for the system. We have also conducted a user study to evaluate our process and templates for identifying security requirements [39]. Results indicate that our process supports the requirements engineering effort by considering multiple security objectives and identifying an initial set of candidate security requirements for the system.

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

[10] http://agile.csc.ncsu.edu/iTrust/wiki/doku.php

TABLE VI. EXAMPLE CONTEXT-SPECIFIC SECURITY REQUIREMENTS TEMPLATES

Security Objective		Security Requirements Templates	Generated Security Requirements
Account-ability	**AY1**	**Logging transactions with sensitive data** *Given:* <subject> = user or role <resource> = sensitive information <action> = create/read/update/delete ***Add Security Requirements:*** • The system shall log every time <subject> [performs the] <action> <on|for> <resource>. [see C1, I4] • At a minimum, the system shall capture the following information for the log entry: <subject> identification, timestamp, <action>, <resource>, and identification of the owner of <resource>. [see IA2, C1, I4]	**Input Sentence:** The system should provide the ability to check medications against a list of drugs noted to be ineffective for the patient in the past. **Security Requirements:** • The system shall log every time user checks medications against a list of drugs noted to be ineffective for the patient in the past. • At a minimum, the system shall capture the following information for the log entry: user identification, timestamp, check medication, patient identification. • The system shall not allow modification of the log by any user.*
Integrity	**I4**	**Maintaining integrity of unchangeable resources** *Given:* <resource> = write-once information (e.g., log files) ***Add Security Requirements:*** • The system shall not allow modification of <resource> by any user. [see AY1]	

* Last requirement is generated based on the template for integrity (I4) as suggested in template AY1. Templates for confidentiality can also be considered.

VIII. THREATS TO VALIDITY

We have considered following threats to validity:

Selection of problem domain: Study oracle created using documents from healthcare domain may not be generalizable to other domains due to different security objectives and domain-specific vocabulary. Moreover, assets that need to be protected are well understood in healthcare domain that may facilitate identification of security-relevant sentences. Many organizations adopt data classification guides that can be used to help guide our process in other domains.

Selection of systems and documents: Security requirements may come from different sources (requirements documents, policy specifications, legislative texts, standards and best practices). 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 identified security-relevant sentences.

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 considered multiple sources from security literature to identify the objectives. A general consensus on the categorization of security objectives minimizes this threat.

Subjective assessment of security objectives: To develop the study oracle, we carried out manual classification of sentences, which can be subjective. Misclassification of sentences based on security objectives in the oracle may have occurred. To minimize this concern, two researchers independently carried out the classification of each document while a third researcher consolidated the final classification. Inter-rater reliability ranges between 0.32 to 0.85, lending validity to the process.

IX. CONCLUSION AND FUTURE WORK

Our work describes a tool-assisted process for identifying key attributes of sentences to be used in security-related analysis and specification of functional security requirements using a set of context-specific templates. We have evaluated our process on six documents from the electronic healthcare domain, identifying 46% of sentences 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 sentences, we extracted 19 context-specific templates and associated reusable functional security requirements. We also provide an oracle of sentences labeled with relevant security objectives for the healthcare domain[11].

To improve the recall of our classification approach and identify security-relevant sentences that may have been missed, we plan to consider features specific to each security objective that may support the classification effort. For instance, we are looking to extract tuples from input sentences that can be used to implement access control. Presence of these tuples can inform the classification for confidentiality and accountability. Identification of such features will also support development of security requirements patterns [40], extending our initial set of context-specific templates. We also plan to evaluate the applicability of our process in domains other than healthcare.

For practitioners, our research can help mitigate security vulnerabilities early in the software development lifecycle by identifying key security requirements that are hidden in plain sight and may otherwise be overlooked.

ACKNOWLEDGEMENT

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.

REFERENCES

[1] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt, "A comparison of security requirements engineering methods,"

[11] http://go.ncsu.edu/securitydiscoverer/

Requirements Engineering - Special Issue on RE'09: Security Requirements Engineering, vol. 15, pp. 7-40, 2010.
[2] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina, "A systematic review of security requirements engineering," *Computer Standards and Interfaces,* vol. 32, p. 13, Jun. 2010.
[3] J. Slankas and L. Williams, "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)*, 2013, pp. 9-16.
[4] D. G. Firesmith, "Engineering security requirements," *J. Object Technology,* vol. 2, p. 16, Jan-Feb. 2003.
[5] D. G. Firesmith, "Specifying reusable security requirements," *Jornal of Object Technology,* vol. 3, p. 15, Jan-Feb. 2004.
[6] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer Standards and Interfaces,* vol. 29, pp. 244-253, Feb 2007.
[7] M. Schumacher, E. Fernandez-Buglioni, D. Hyberston, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*. West Sussex: John Wiley & Sons, Ltd, 2006.
[8] 2012, Common Criteria for Information Technology Security Evaluation,Version 3.1. Release 4. Available: http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf
[9] D. Firesmith, "Specifying reusable security requirements," *Jornal of Object Technology,* vol. 3, p. 15, Jan-Feb. 2004.
[10] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina, "A systematic review of security requirements engineering," *Computer Standards & Interfaces,* vol. 32, pp. 153-165, 2010.
[11] M. Howard and S. Lipner, *The Security Development Lifecycle*. Redmond, WA: Microsoft Press, 2006.
[12] N. R. Mead, E. D. Houg, and T. R. Stehney, "Security Quality Requirements Engineering (SQUARE) Methodology," Software Engineering Inst., Carnegie Mellon University2005.
[13] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering,* vol. 10, p. 12, 2005.
[14] A. v. Lamsweerde, "Elaborating security requirements by construction of intentional anti-Models," presented at the International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland, 2004.
[15] V. N. L. Franqueira, T. T. Tun, Y. Yu, R. Wieringa, and B. Nuseibeh, "Risk and argument: A risk-based argumentation method for practical security," presented at the IEEE International Requirements Engineering Conference, 2011.
[16] P. Salini and S. Kanmani, "Survey and analysis on security requirements engineering," *Computers and Electrical Engineering,* vol. 38, p. 13, 2012.
[17] Kurt Schneider, Eric Knauss, Siv Houmb, Shareeful Islam, and J. Jürjens, "Enhancing security requirements engineering by organizational learning," *Requirements Engineering,* vol. 17, pp. 35-56, 2012.
[18] T. D. Breaux and A. I. Antón, "Analyzing regulatory rules for privacy and security requirements," *IEEE Transactions on Software Engineering,* vol. 34, pp. 5-20, Jan. 2008.
[19] J. C. Maxwell, A. I. Antón, and P. Swire, "A legal cross-references taxonomy for identifying conflicting software requirements," in *Requirements Engineering*, 2011, pp. 197-206.
[20] Q. He and A. I. Antón, "Requirements-based Access Control Analysis and Policy Specification (ReCAPS)," *Information and Software Technology,* vol. 51, pp. 993-1009, 2009.
[21] X. Xiao, A. Paradkar, S. Thummalapenta, and T. Xie, "Automated extraction of security policies from natural-language software documents," in *International Symposium on the Foundations of Software Engineering (FSE)*, ed. Raleigh, North Carolina, USA, 2012.
[22] G. Salton and M. J. McGill, "Introduction to Modern Information Retrieval," 1986.
[23] J. Cleland-Huang, R. Settimi, and P. Solc, "The detection and classification of non-functional requirements with application to early aspects," in *14th IEEE International Requirements Engineering Conference (RE'06)*, IEEE, 2006, pp. 39-48.
[24] A. Casamayor, D. Godoy, and M. Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," *Information and Software Technology,* vol. 52, pp. 436-445, 2010.
[25] W. Zhang, Y. Yang, Q. Wang, and F. Shu, "An empirical study on classification of non-functional requirements," in *The Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, 2011, pp. 190-195.
[26] 2013, Security and Privacy Controls for Federal Information Systems and Organizations. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf
[27] 2001, Underlying Technical Models for Information Technology Security. Available: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf
[28] 2004, Standards for Security Categorization of Federal Information and Information Systems. Available: http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf
[29] 2006, Minimum Security Requirements for Federal Information and Information Systems Available: http://csrc.nist.gov/publications/fips/fips200/FIPS-200-final-march.pdf
[30] 2002, Federal Information Security Management Act. Available: http://csrc.nist.gov/drivers/documents/FISMA-final.pdf
[31] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Communication of the ACM,* vol. 17, 1974.
[32] 1990, IEEE Standard Glossary of Software Engineering Terminology. Available: http://standards.ieee.org/findstds/standard/610.12-1990.html
[33] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady,* vol. 10, pp. 707-710, 1966.
[34] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics,* vol. 33, pp. 159-174, 1977.
[35] M. Hall, H. National, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software : An Update," *SIGKDD Explorations,* vol. 11, pp. 10-18, 2009.
[36] J. Han, M. Kamber, and J. Pei, "Data Mining: Concepts and Techniques," p. 744, 2011.
[37] J. R. Quinlan, "Induction of decision trees," *Machine Learning,* vol. 1, 1986.
[38] Y. Yang and P. J.P., "A comparative study on feature selection in text categorization " in *Fourteenth International Conference on Machine Learning (ICML'97)*, 1997, pp. 412-420.
[39] M. Riaz, J. Slankas, J. King, and L. Williams, "Using templates to elicit implied security requirements from functional requirements − A controlled experiment," to present at the International Symposium on Empirical Software Engineering and Measurement (ESEM), Torino, Italy, 2014.
[40] S. Withall, *Software Requirement Patterns.*: O'Reillly, 2007.