Using Templates to Elicit Implied Security Requirements from Functional Requirements − A Controlled Experiment

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

ABSTRACT

Context: Security requirements for software systems can be challenging to identify and are often overlooked during the requirements engineering process. Existing functional requirements of a system can imply the need for security requirements. Systems having similar security objectives (e.g., confidentiality) often also share security requirements that can be captured in the form of reusable templates and instantiated in the context of a system to specify security requirements.
Goal: We seek to improve the security requirements elicitation process by automatically suggesting appropriate security requirement templates implied by existing functional requirements.
Method: We conducted a controlled experiment involving 50 graduate students enrolled in a software security course to evaluate the use of automatically-suggested templates in eliciting implied security requirements. Participants were divided into treatment (automatically-suggested templates) and control groups (no templates provided).
Results: Participants using our templates identified 42% of all the implied security requirements in the oracle as compared to the control group, which identified only 16% of the implied security requirements. Template usage increased the efficiency of security requirements identified per unit of time.
Conclusion: Automatically-suggested templates helped participants (security non-experts) think about security implications for the software system and consider more security requirements than they would have otherwise. We found that participants need more incentive than just a participatory grade when completing the task. Further, we recommend to ensure task completeness, participants either need a step-driven (i.e., wizard) approach or progress indicators to identify remaining work.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications – *Elicitation Methods.*

General Terms

Measurement, Experimentation, Security, Human Factors.

Keywords

Security Requirements, Templates, Controlled Experiment.

1. INTRODUCTION

Security requirements of software systems can be challenging to identify and are often overlooked given the lack of focus on security during early stages of system development [5]. Several approaches exist to support security requirements engineering by providing guidelines for the identification of security objectives, important resources and assets, and associated security threats to a software system. However, these existing approaches still require a sufficient level of security expertise and awareness to understand the security objectives of a system and to specify security requirements to meet those objectives. Feedback from practitioners across various organizations emphasize that a tool-assisted process to aid in security requirements elicitation may facilitate secure software development [16]. By automating parts of the security requirements engineering process, we can enhance the ability of security non-experts to actively consider security concerns and reduce the amount of manual effort.

Systems that share common security objectives, such as confidentiality and integrity, often have similar security requirements [6], thus providing an opportunity to capture such common requirements in the form of reusable templates that can be instantiated in the context of a system. We can support the requirements engineering effort for both security experts and non-experts by identifying a core set of reusable security requirements templates and automatically suggesting applicable templates. Existing functional requirements of a system can imply the need for security requirements. Natural language processing can be used to identify the patterns in functional requirements that have security implications [15]. These patterns can also lead to the choice of a requirements template.

We seek to improve the security requirements elicitation process by automatically suggesting appropriate security requirement templates implied by existing functional requirements.

In previous work [15], we developed a tool-assisted process that takes as input a set of requirements-related natural language artifacts (e.g., use case-based requirements specifications) and suggests security requirements templates that may apply based on statements in the input artifact. We identify security objectives implied by the statements and suggest templates that help translate the security objectives into explicit sets of security functional requirements. In the context of this research, statements with security implications are statements suggesting a need for security. For example, the sentence "*Health care professionals*

(HCPs) can return to an office visit and modify or delete the fields of the office visit." implies security objectives of confidentiality (*data for the office visit should be only viewable by HCPs*), integrity (*when reading, writing or deleting fields of office visit*), and accountability (*logging transactions with sensitive data*). Based on the identified objectives, we suggest the appropriate security requirement templates. Table 1 shows the template '*C1: maintaining confidentiality of data*' and the concrete security requirements that are generated by instantiating the template based on the example statement.

In this paper, we report the results of a controlled experiment, a user study of 50 graduate students enrolled in a software security course at North Carolina State University (NCSU), to evaluate the use of our automatically-suggested templates in eliciting security requirements for the system under consideration. Specifically, we analyze the following research questions (RQ):

RQ1: What is the **quality** of security requirements elicited through the use of automatically-suggested security requirements templates?
RQ2: How **complete** are the security requirements elicited through the use of automatically-suggested security requirements templates?
RQ3: How **relevant** are the security requirements elicited through the use of automatically-suggested security requirements templates?
RQ4: How **efficient** is the process of eliciting security requirements through the use of automatically-suggested security requirements templates?

Our study participants received use case scenarios from the electronic healthcare software domain and were tasked to document relevant security requirements.

Our work makes the following major contributions:

- An empirical evaluation of the use of security requirements templates in security requirements elicitation.
- Availability of detailed study material and data to support future experimentation, replication and analysis.
- Lessons learned on how to improve security requirements elicitation effort in an experimental setting

Table 1. Security requirements template '*C1: maintaining confidentiality of data*' and concrete requirements

Security Require-ments Template	• The system shall enforce access privileges that <enable|prevent> <subject> to <action> <resource>. • The system shall encrypt <resource> and store <resource> in encrypted format using an industry-approved encryption algorithm. • The system shall transmit <resource> data in encrypted format to and from the authorized <subject>. • The system shall monitor the status and location of system components that may contain unencrypted <resource> data.
Concrete Security Require-ments	• The system shall enforce access privileges that **enable HCPs** to **modify or delete the fields of office visit**. • The system shall encrypt **office visit** and store **office visit** in encrypted format using an industry-approved encryption algorithm. • The system shall transmit **office visit** data in encrypted format to and from the authorized **HCPs**. • The system shall monitor the status and location of system components that may contain unencrypted **office visit** data.

Rest of this paper is organized as follows: we present an overview of our tool-assisted process in Section 2. In Section 3, we provide details of our research methodology. In Section 4, we present the results of our experiment and discussion. In Section 5, we present the threats to validity that we have considered. In Section 6, we discuss the lessons learned. Section 7 presents background and related work. We conclude the paper in Section 8.

2. SECURITY DISCOVERER PROCESS

In prior work [15], we developed a tool-assisted process for identifying implied security requirements from natural language project artifacts. Our process takes as input requirements-related artifacts (requirement specifications, feature requests, use case scenarios etc.) and generates a set of security requirements as output. We automatically parse individual statements in input artifacts and identify which (if any) security objectives relate to each statement. Table 2 lists security objectives considered in our process.

Table 2. Security objectives

Security Objective	Description
Confidentiality (C)	The degree to which the "data is disclosed only as intended". [17]
Integrity (I)	The degree to which a system or component guards against improper modification or destruction of computer programs or data. [3]
Availability (A)	"The degree to which a system or component is operational and accessible when required for use." [1]
Identification & Authentication (IA)	The need to establish that "a claimed identity is valid" for a user, process or device. [2]
Accountability (AY)	The degree to which actions affecting software assets "can be traced to the actor responsible for the action." [17]
Privacy (PR)	The degree to which "an actor can understand and control how their information is used." [15]

We evaluated the precision and recall of our tool-assisted process in identifying security objectives, implied by natural language statements, in six documents from electronic healthcare domain. We correctly predicted 82% of the security objectives for all the statements in input artifacts (precision). We also identified 79% of all security objectives implied by the statements within the input artifacts (recall) [15].

Based on the identified objectives, we automatically present a list of applicable security requirements templates that can be selected and instantiated by a requirements engineer with details from the initial statement using the authoring mechanism provided in the tool. We also support traceability of generated security requirements back to the source statements in the input artifacts.

3. RESEARCH METHODOLOGY

We present our methodology for conducting the experiment in accordance with the reporting guidelines by Jedlitschka et al. [7]. We used the quality checklist for quantitative studies by Kitchenham et al., [8] to guide our research design and execution.

3.1 Goals, Hypotheses and Metrics

The goal of this experiment is to determine whether the use of automatically-suggested security requirements templates leads to efficient and effective requirements elicitation when compared to a manual approach based on personal expertise. We test the following null hypotheses in this experiment to address our research questions:

H₀₁: The **quality** of elicited security requirements is unrelated to the use of automatically-suggested security requirements templates.

H₀₂: The **quantity** of elicited security requirements is unrelated to the use of automatically-suggested security requirements templates.

H₀₃: The **relevance** of elicited security requirements is unrelated to the use of automatically-suggested security requirements templates.

H₀₄: The **efficiency** of the requirements elicitation process is unrelated to the use of automatically-suggested security requirements templates.

We use the metrics listed in Table 3 to test the preceding hypotheses. The outcomes of the study (i.e., security requirements identified in each participant's response) are evaluated in terms of the given metrics, as explained in Section 4. Based on responses to a post-study survey (Section 3.3), we compare the templates and non-templates groups to identify interesting feedback, suggestions for improvement, and general reflections on the study.

Table 3. Metrics used for evaluating participants' responses

Evaluation Criteria	Metric Type	Metrics Used
Quality, *of security requirements*	Qualitative	Likert-like scale (1-5): lower score indicates lower quality
Quantity, *of security requirements*	Quantitative	Requirements coverage score: Percentage of security requirements covered in the oracle
Relevance, *of security requirements*	Quantitative	Relevance ratio: Ratio of relevant requirements to total requirements identified
Efficiency, *of eliciting requirements*	Quantitative	# of correct requirements identified per unit of time

3.2 Participants

Our study participants were graduate students enrolled in a 16-weeks software security course[1] offered at NCSU. We conducted the study as an online web-based activity during the last week of the course, after students had already learned various software security concepts. The task for this study was mandatory for all the students to complete, similar to other class exercises. However, students could opt-out of participating in the study[2], which would preclude the inclusion of their work in the study results. Of the 54 students enrolled in the course, 50 gave consent to use their responses for the study. Each student received coursework credit for completing the task as a classroom exercise, irrespective of their decision to participate in the study or of the quality of their responses.

We automatically assigned study participants (students who agreed to participate in the study) to one of two experiment groups in a round-robin fashion: a) treatment group, and b) control group. Both groups were given the task of identifying security requirements. However, the treatment group was assisted with automatically-suggested security requirements templates. The control group performed the same task without the assistance of suggested templates. Both groups received specific sets of instructions, reference material, and task screens (Section 3.4).

At the end of the task for this study, we retained the data from only those students who gave consent to participate. We recorded no personally identifiable information about the participants (e.g., name, student identifier). Table 4 summarizes the participant's background. Participants in the treatment and control groups were similar in terms of academic experience in computer science (CS, majority having > 5 years), software engineering (SE, majority having 3-5 years), and security (Sec, majority having < one year).

Table 4. Participants experience frequency
(CS: ***Computer Science;*** **SE:** ***Software Engineering;*** **Sec:** ***Security*****)**

Group	Experience (yrs)	Academic			Work Experience		
		CS	SE	Sec	CS	SE	Sec
Treatm-ent	> 5 years	16	4	2	0	0	0
	3-5 years	9	11	2	11	6	1
	1-2 years	1	7	7	4	5	2
	<1 year	0	4	15	11	15	23
	No Response	4	4	4	4	4	4
Control	> 5 years	11	3	0	0	0	0
	3-5 years	8	10	2	9	7	0
	1-2 years	0	4	5	3	6	5
	<1 year	1	3	13	8	7	15
	No Response	0	0	0	0	0	0

The only area of expertise in which participants differed between the treatment and control group was work experience in SE. The majority of participants in the treatment group had less than one year of SE experience, whereas participants in the control group were evenly divided into three experience categories (3-5 year, 1-2 years, <1 year). Almost all participants had less than one year of work experience related to security. Similarities in background and expertise of participants indicate that differences in outcome are unrelated to a participants' experience.

3.3 Study Environment

All students received a URL to access the online site for the study. The system first showed students the consent form. Students then read the consent form and could either allow or deny use of their data in the study results. Next, the system assigned each student an auto-generated random access code. Students could save the task at any point during the experiment and return to the task by entering their access code at the provided URL. Students were encouraged to work on the task during the 60-minute lecture period in a classroom setting. We provided a five minute overview of the task at the beginning of the lecture period. In addition to the remaining 55 minutes, students had a total of two days to complete the task and required to submit the task before the start

[1] http://go.ncsu.edu/csc-591-software-security

[2] The study was approved by the NCSU IRB (3567) and the ARO IRB (Proposal Log Number 63141-CS, Award Number W911NF-13-1-0094, HRPO Log Number A-17945.2).

of the next lecture period. Of the 50 participants, 40 completed the task during the lecture period.

After completing the task, we solicited feedback on the process used to perform the task by asking participants to:

- Briefly explain the process used for identifying applicable security requirements (e.g., what information you looked at in the use case or reference material).

For the participants in the treatment group, we asked the following additional open-ended questions:

- What is your opinion regarding use of requirements templates?
- What is your opinion regarding use of generated requirements?

For each participant, we recorded total time spent completing the task and whether the participant submitted the task. All students in the study submitted the task.

3.4 Experiment Material

The experiment material consists of the reference material given to the participants, as well as the use case scenarios on which the task had to be performed. The reference material, use cases and other study documents are available on our project website[3].

3.4.1 *Reference Material*

All participants received four pages of reference material containing a description of software security objectives (see Section 2) as well as textual clues that can indicate an implied security objective. Reference material also contained a total of 40 example security requirements grouped by security objectives. We provided this standard reference material two days prior to the start of the experiment in the classroom. During the experiment, the control group had access to the same reference material online. However, for the treatment group, we presented example security requirements in two forms: i) 16 reusable security requirements templates grouped by security objectives, and ii) 40 concrete example security requirements (also available to control group) that were generated from the templates. We list names of the security requirements templates below, grouped by security objective. Detailed templates are available online[3].

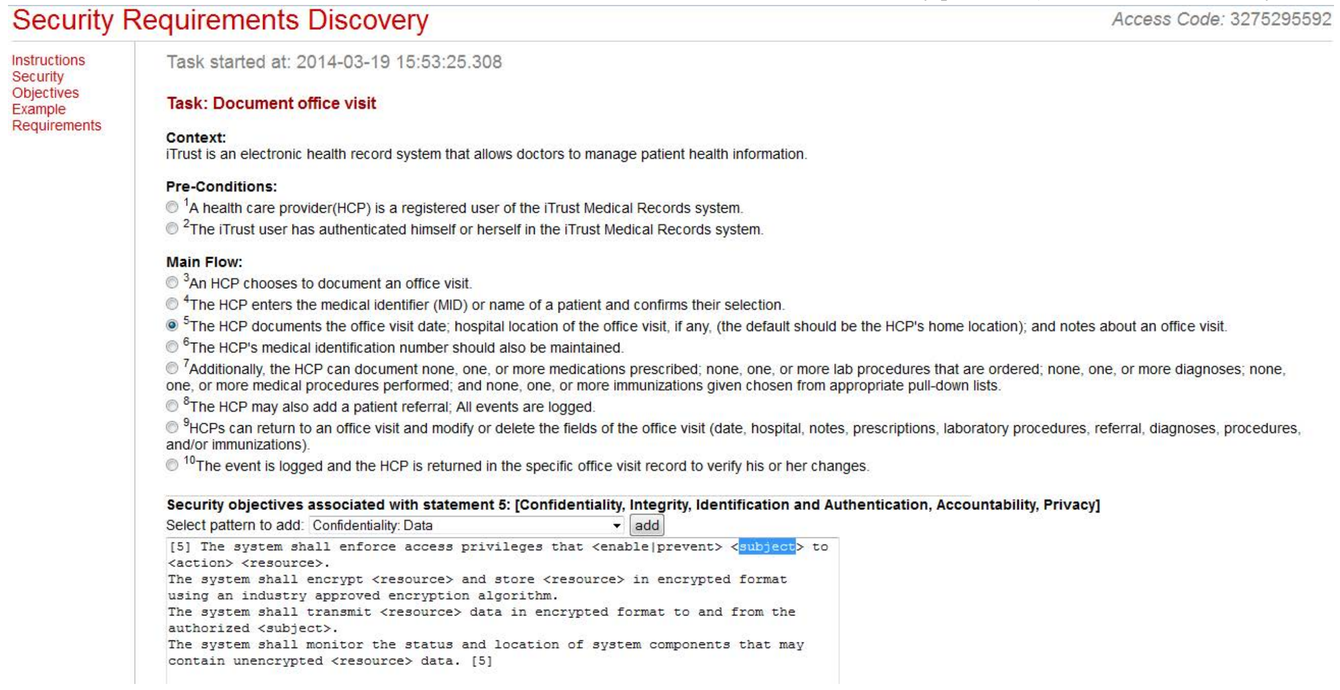
- *Confidentiality*: *C1*-maintaining confidentiality of data;
- *Integrity*: *I1*- read-type actions; *I2*- write-type actions; *I3*-delete actions; *I4*-unchangeable resources;
- *Availability*: *A1*-availability of data; *A2*-appropriate response time; *A3*-service availability; *A4*-backup and recovery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

3.4.2 *Use cases*

Security is an important consideration in a number of domains, including healthcare. We selected two use cases, both from the electronic healthcare domain that met the following criteria:

- Focus on a single unit of functionality, such that participants could easily understand the scope of the requirements.
- Require no understanding of domain-specific taxonomies.
- Imply at least four different types of security objectives.
- Use case specifications openly accessible.

First use case (UC1 - Document office visit) is from the iTrust[4] electronic health record (EHR) system [14], an open-source system developed by students at NCSU. Second use case (UC2 - Retrieve exam results by patient ID) is based on a user story[5] from



Task started at: 2014-03-19 15:53:25.308

Task: Document office visit

Context:
iTrust is an electronic health record system that allows doctors to manage patient health information.

Pre-Conditions:

○ ^{1}A health care provider(HCP) is a registered user of the iTrust Medical Records system.
○ ^{2}The iTrust user has authenticated himself or herself in the iTrust Medical Records system.

Main Flow:

○ ^{3}An HCP chooses to document an office visit.
○ ^{4}The HCP enters the medical identifier (MID) or name of a patient and confirms their selection.
● ^{5}The HCP documents the office visit date; hospital location of the office visit, if any, (the default should be the HCP's home location); and notes about an office visit.
○ ^{6}The HCP's medical identification number should also be maintained.
○ ^{7}Additionally, the HCP can document none, one, or more medications prescribed; none, one, or more lab procedures that are ordered; none, one, or more diagnoses; none, one, or more medical procedures performed; and none, one, or more immunizations given chosen from appropriate pull-down lists.
○ ^{8}The HCP may also add a patient referral; All events are logged.
○ ^{9}HCPs can return to an office visit and modify or delete the fields of the office visit (date, hospital, notes, prescriptions, laboratory procedures, referral, diagnoses, procedures, and/or immunizations).
○ ^{10}The event is logged and the HCP is returned in the specific office visit record to verify his or her changes.

Security objectives associated with statement 5: [Confidentiality, Integrity, Identification and Authentication, Accountability, Privacy]

Select pattern to add: Confidentiality: Data

```
[5] The system shall enforce access privileges that <enable|prevent> <subject> to
<action> <resource>.
The system shall encrypt <resource> and store <resource> in encrypted format
using an industry approved encryption algorithm.
The system shall transmit <resource> data in encrypted format to and from the
authorized <subject>.
The system shall monitor the status and location of system components that may
contain unencrypted <resource> data. [5]
```

Figure 1. Task screen for treatment group.

[3] http://go.ncsu.edu/secreqtemplatesstudy

[4] http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=requirements

[5] http://www.va.gov/vler/vlerdocs_userstories.asp

Virtual Lifetime Electronic Record (VLER), a business and technology initiative that allows secure and standardized electronic exchange of health and benefits information for United States Veterans and Service members.

We numbered all statements in each use case and provided a brief context statement to help participants understand the scope and purpose of the system for which the security requirements have to be identified. During the study, participants in the treatment group could select any of the statements in the given use case scenario by clicking on the corresponding radio button, see the security objectives implied by that statement, and receive automated suggestions of applicable templates, as shown in Figure 1. Once a participant selected a template, the template was copied in the text area and could be manually filled-in with details (such as subject, action, resource) from the original statement in the use case. Participants could select the statements in any order. Participants in the control group had no knowledge of the templates, nor did control group participants receive any suggestions for applicable security requirements. Control group participants could use example security requirements from the reference material to identify, formulate, and document similar requirements. Participants could save or submit the task at any time.

3.4.3 *Oracle of Security Requirements*

We created an oracle of the security requirements for each use case to evaluate the relevance and completeness of requirements identified by the participants. Five security researchers, including the first three authors, participated in developing the oracle before the execution of the study. The steps for creating the oracle are:

- For each statement in the use case, identify the security objectives associated with the statement.
- For each identified objective, select the security requirements templates that are applicable.
- For each applicable security requirements template, fill-in the templates with contextual details from the original statement to generate concrete security requirements.
- Remove duplicate or redundant requirements.

We used Delphi method [12] to identify security objectives associated with each statement. For each statement and each objective, five researchers independently voted either 'yes' or 'no' and then revealed their votes simultaneously. After each vote, researchers reasoned about their choice and voted again, until all researchers voted either 'yes' or 'no' for the objective under consideration. We provide a statement-wise breakdown of applicable security requirements templates in Table 5.

Table 5. Templates associated with use case statements

Use Case	# of Statements	Applicable Templates per Statement

UC1	10	**1**: IA2;
		2: IA2; IA3-1; AY2;
		3,4,5, 7,8: C1; I2; IA2; AY1; PR1;
		6: I2; IA2; AY1;
		9: C1; I1-3; A1-A; A2-A; IA2; AY1; PR1;
		10: C1; I1; IA2; AY1;

UC2	9	**1**: C1; PR1;
		2: AY1;
		3,4,5,8: C1; AY1; PR1;
		6: C1; AY1;
		7: None;
		9: C1; A1-A; A2-A; AY1; PR1;

We identified 110 security requirements for UC1 and 35 security requirements for UC2 as part of the oracle. Although both use cases had a similar number of statements, UC1 involves interactions with multiple resources and, consequently, a higher number of security requirements.

3.5 Experiment Design

We used a 2x2 between-subject design for the purpose of this study [11]. The participants were divided into four groups based on the process used for identifying requirements (automatically-suggested security requirements templates vs. no templates) and the use case assigned (UC1 or UC2). To minimize potential bias, participants did not know about the existence of different groups or use cases. They were just informed that they will be given a use case scenario and will have to identify security requirements from the scenario in accordance with the instructions. For traceability, participants entered the security requirements in a text area followed by statement number(s) from use case scenario to which the security requirement relates. Table 6 provides the number of participants in treatment (provided with automatically-suggested security requirements templates) and control (no security requirements templates provided) groups for both use cases.

Table 6. Number of participants in each group

Requirements Process	Use Case		Total
	UC1	UC2	
Treatment	16	14	**30**
Control	10	10	**20**
Total	**26**	**24**	**50**

4. RESULTS AND ANALYSIS

We next present the results of our experiment, followed by our data analysis to address the research questions. We conducted a 2x2 ANOVA (available online[3]) for unbalanced groups, adjusting for multiple comparisons, to test the four null hypotheses.

4.1 RQ1: Quality of Identified Requirements

We evaluated the overall quality of the outcome, i.e., identified security requirements, on a Likert-like scale of 1 to 5 (1: poor; 2: below average; 3: average; 4: above average; 5: good). Two researchers independently assigned quality scores for each outcome. The following questions guided the scoring process:

- Are the requirements too general or too specific?
- Have all necessary elements of the requirements been identified (e.g., subject, object, resource, and data to be logged)?
- Are there any logical inconsistencies in the requirements?
- For the treatment group, have the selected templates been filled-in with appropriate details?

We applied weighted kappa [18] using linear weights to account for how far apart the two raters were in assigning the quality score. The weighted kappa score was 0.588 indicating that the strength of agreement between the two raters was 'moderate' [10]. Using a paired t-test, we found no statistically significant difference between the scores assigned by the two raters at $p<0.05$. We use the average of quality scores assigned by both raters for subsequent analysis.

4.1.1 Analysis

We found no statistically significant difference between the quality of requirements identified by the treatment vs. control groups (p-value=0.928) or between either of the use cases (p-value=0.891) at $p<0.05$. Thus, we fail to reject the null hypothesis H_{01} and cannot state that the quality of identified requirements is affected by the use of security requirements templates. The mean quality scores for each group are given in Table 7.

Table 7. Mean quality scores for each group

Requirements Process	Use Case		Overall
	UC1	UC2	
Treatment	2.78	2.96	**2.87**
Control	2.95	2.85	**2.90**
Overall	**2.87**	**2.91**	**2.88**

4.1.2 Discussion

Overall, neither group produced quality requirements. Although the treatment group was able to identify the applicable requirements templates, one-third of the participants did not fill-in the templates (8 out of 16 for UC1; 2 out of 14 for UC2). In addition to the classroom time, participants had two days to complete the exercise. Only one participant who started the task in class submitted it later, after briefly reviewing the response. Since study responses were collected anonymously, participants were required to submit a separate sheet of paper to the instructor to receive credit for participation in the classroom exercise (irrespective of their participation in the study). Anonymity, combined with a lack of a stronger incentive, might have reduced participant motivation for producing a top quality assignment.

Only one participant in the treatment group did not use the automatically-suggested templates. Conversely, many participants in the control group incorporated quality security requirements by leveraging the example requirements provided as part of the reference material. The control group participants who did not use the reference material often documented requirements that were too general (e.g., "Send the results over a secure channel") or too specific, discussing security mechanisms (e.g., "Avoid URL jumping by using HTTP referrer fields...").

Providing example security requirements, organized by security objectives, improved the quality of generated security requirements for both the treatment and the control groups.

4.2 RQ2: Quantity of Identified Requirements

We used the following 'coverage score' to compute the quantity of identified requirements in each participant's response:

- *Requirements coverage score*: Percentage of total requirements in the oracle that are covered by the requirements in the participant's response.

We compute the coverage score, instead of simply counting the number of requirements in a participant's response, because we do not expect a one-to-one mapping of security requirements in participants' responses and the oracle. One requirement in a response could map to multiple requirements in the oracle or vice versa. No participant suggested any additional correct security requirements that were not already part of the oracle.

4.2.1 Analysis

We found both the requirements process (treatment vs. control, p-value=<0.0001) and use case (UC1 or UC2, p-value=0.0002) to be significant factors in determining the quantity of security requirements at $p<0.05$. The interaction between the requirements process and use case was also significant (p-value=0.0054). Thus, we reject the null hypothesis H_{02} and determine that the use of automatically-suggested security requirements templates helped in identifying significantly more requirements when compared to the control group. Mean coverage percentages are given in Table 8.

Table 8. Mean coverage scores (%) for each group

Requirements Process	Use Case		Overall
	UC1	UC2	
Treatment	24.26%	61.63%	**42.95%**
Control	12.45%	18.57%	**15.51%**
Overall	**18.36%**	**40.1%**	**31.22%**

Overall participants identified only 31% of all the security requirements in the oracle. Participants in the treatment group identified significantly more requirements when compared to the participants in the control group. For UC1, participants in treatment group identified twice as many requirements as control group. For UC2, treatment group identified three times the requirements identified by the control group. Table 9. provides a breakdown of security requirements identified for each objective.

In the treatment group for UC1, participants identified some requirements related to the security objectives of identification & authentication, accountability, confidentiality and privacy. In both groups for UC1, almost all participants failed to identify any of the 44 integrity-related requirements. Of these, 23 integrity-related requirements were not identified by any participant. The integrity of sensitive information can be compromised if these requirements are not considered.

Participants in the treatment group for UC2 identified some requirements related to the objectives of confidentiality, privacy

Table 9. Breakdown of requirements by security objectives

Use Case	Security Objective *(Table 2)*	Requirements in oracle for this objective # (% of total)	Requirements identified for each objective on average # (% covered)	
			Treatment	Control
UC1	C	22 (20%)	8 (36%)	4 (16%)
	I	44 (40%)	0 (0%)	2 (4%)
	A	1 (1%)	0 (0%)	0 (0%)
	IA	2 (2%)	1 (50%)	1 (50%)
	AY	20 (18%)	11 (55%)	7 (35%)
	PR	21 (19%)	7 (33%)	0 (0%)
	TOTAL	**110**	**27 (24.3%)**	**14 (12.5%)**
UC2	C	15 (43%)	10 (66%)	6 (38%)
	A	2 (6%)	0 (0%)	0 (0%)
	AY	12 (34%)	7 (58%)	0 (0%)
	PR	6 (17%)	5 (80%)	0 (0%)
	TOTAL	**35**	**22 (61.6%)**	**6 (18.6%)**

and accountability. The treatment group for UC2 identified almost 62% of all requirements on average. For the control group for UC2, participants identified confidentiality related requirements but failed to identify requirements for privacy and accountability.

4.2.2 Discussion

We found significant differences in the quantity of identified requirements in the treatment versus control groups, as well as for the two use cases given to the participants. Automatically-suggested templates helped participants in the treatment group to consider more security requirements for the system. Participants in the control group did not know about the templates. Although reference material for the control group contained information about the security objectives and various textual clues that could imply an objective, participants had to figure out applicable objectives on their own. Participants in the control group were not able to incorporate requirements to meet the various security objectives of the system indicating that they were not able to identify and consider all the security objectives.

Knowledge of security objectives and availability of applicable requirements templates leads to significantly higher coverage of security requirements for the system overall. Participants may not be able to consider all the implied security objectives, resulting in incomplete security requirements identification and potential gaps in security-related functionality of the system.

4.3 RQ3: Relevance of Identified Requirements

To assess whether the time spent on identifying security requirements led to identification of requirements relevant to the given scenario or not, we consider the ratio of relevant requirements to total requirements in each response. If a participant identified only one requirement, which is relevant, they will have 100% ratio of relevant to total requirements identified. Ratio of relevant requirements is analogous to accuracy of responses. Total number of correct requirements identified compared to the study oracle is already reflected by the coverage score discussed in Section 4.2, which is analogous to recall. If a participant identified a confidentiality requirement for statement # X, and the statement implied a need for confidentiality, then the identified requirement is relevant even if it is not adequately specified. Thus the measure for relevance is independent from the measure for quality.

4.3.1 Analysis

We did not find statistically significant difference in the ratio of relevant requirements in the response between treatment and control groups (p-value = 0.34) at $p<0.05$. Thus, we fail to reject the null hypothesis H_{03} that ratio of relevant requirements to total requirements identified is unrelated to the use of security requirements templates. The mean ratio (shown as percentage) of relevant requirements is given in Table 10.

Although we did not find a statistically significant difference in the ratio of relevant requirements identified between various groups, participants in the treatment group performed slightly better (90% of the identified requirements were relevant) than the control group (85% of the identified requirements were relevant).

Table 10. Mean ratio of relevant to total requirements for each group (%)

Requirements Process	Use Case		Overall
	UC1	UC2	
Treatment	90	91	**90**
Control	86	84	**85**
Overall	**88**	**87**	**88**

4.3.2 Discussion

Participants in the treatment group received suggestions related to the security requirements templates that might be applicable based on the security objectives implied by statements in the given use case. However, not all the suggested templates may be applicable for a given functional requirement. Some of the participants in the treatment group selected all suggested templates increasing the number of irrelevant requirements identified. Overall, participants in both treatment and control group listed mostly relevant security requirements (over 88% of all the identified requirements were relevant).

Although participants identified only 31% of the applicable security requirements on average (see Table 8), the identified requirements were mostly relevant.

4.4 RQ4: Efficiency of Requirements Elicitation Process

We evaluated the efficiency of the requirements elicitation process through an analysis of the number of requirements identified per unit of time. We computed the time spent on task, starting from the time a participant accessed the online task until the time participant submitted the task. We also recorded the time when the participants would save the task and then resume the task later on. Only one participant completed the task in multiple sittings, spending only a short time reviewing the task before submitting.

4.4.1 Analysis

We found both the requirements process (treatment vs. control, p-value=<0.038) and use case (UC1 or UC2, p-value=0.01) to be significant factors in determining the efficiency at $p<0.05$. Thus, we reject the null hypothesis H_{04} and determine that the use of our automatically-suggested templates significantly improved the efficiency of the requirements elicitation process. Mean efficiency of each group is given in Table 11.

Table 11. Mean efficiency for each group (# of requirements identified per minute)

Requirements Process	Use Case		Overall
	UC1	UC2	
Treatment	1.7	1	**1.35**
Control	1.15	0.4	**0.77**
Overall	**1.43**	**0.7**	**1.14**

Participants in the treatment group spent an average of six additional minutes eliciting security requirements (22 minutes for treatment vs. 16 minutes for control). Participants spent an average of 17 minutes for UC1 vs. 21 minutes for UC2. On average, participants spent 20 minutes on the task and differences in mean time on task between various groups are not statistically significant. We found a strong linear relation between task time and requirements identified (Pearson correlation coefficient [4] = 0.44). Participants who spent more time on the task identified more security requirements in general.

4.4.2 Discussion

Participants in the treatment group identified twice as many security requirements per unit of time when compared to the control group. Participants working on UC1 were also twice as efficient in identifying security requirements as compared to UC2. This can be attributed to the fact that UC1 has three-times as many security requirements as UC2 so there were more requirements to be considered. The lack of significant difference in mean time on task between various groups can be attributed to the observation that the participants perceived the study as a classroom activity and did not use extra time to work on the task.

Participants in the treatment group identified twice as many security requirements in the same amount of time as compared to the control group, indicating that our process is efficient in terms of the number of security requirements identified per unit of time.

4.5 Self-Reported Solution Approach

We asked the participants to summarize the approach they used for identifying security requirements in a post-task survey.

In the treatment group, all the participants used suggested templates to formulate the security requirements, with one exception. The only participant who did not use the templates indicated a methodology based on identifying possible attacks. Participants in the treatment group reported following approaches to formulating security requirements:

- *Considered suggested templates*: think how templates map to scenario; identify keywords and key phrases; identify resources, action and users; fill in the templates;
- *Considered security objectives*: keep objectives at the back of mind; see if objective might be violated somewhere;
- *Individual expertise*: use intuition to guide the analysis; familiarity with the given scenario; consider potential risks in that scenario; consider assets that need to be protected; think like an attacker, what an attacker can gain;

In terms of individual expertise, one participant reported using an approach based on intuition to identify security requirements. Two participants reported that they were familiar with the system from which the use case scenario was used in the study, so they leveraged their background and experience with the system. However, the two participants who were familiar with the system identified less than the average number of security requirements in their group, indicating that experience did not help.

Participants in the control group reported more diverse approaches to identify security requirements, as listed below:

- *Keywords*: identify keywords from the statements that indicate a need for security (e.g., logging, sends data, process data, identification and authentication)
- *Identify security objectives*: identify applicable security objectives from the given scenario;
- *Example security requirements*: identify relevant examples of security requirements from the reference material; try to map example security requirements to the given scenario;
- *Individual expertise*: identify standard issues; security requirements and methodologies taught in the course (e.g., misuse cases, secure entry and exit points, secure the data);

Many participants in the control group seemed to manually iterate through steps similar to the automated process provided for the treatment group, such as identifying security objectives from statements and trying to map relevant example security requirements to the given scenario. Many participants also relied on their background knowledge and material covered during the course to help guide their analysis.

4.6 Feedback on Security Requirements Templates

We solicited feedback from participants in the treatment group on the use of the security requirements templates and the generated security requirements in a post-task survey. The feedback was voluntary. Four participants did not fill out the post-task survey.

The majority of the responses from participants highlight that participants found the templates and generated requirements helpful. The templates helped participants think about security and provided a starting point to identify relevant security requirements. However, the templates should not be viewed as an exhaustive list, as pointed out by some of the participants. We summarize feedback related to the templates below:

- *Provide a good starting point*: provide a direction for developing secure systems; can use the templates and add additional statements if needed; easy to start defining security requirements with the templates;
- *Help in thinking about security*: allows users with minimal knowledge to use and understand the templates; helped in thinking about context of the statements in the use case; helped in considering more security requirements than would have otherwise;
- *Help in phrasing requirements*: saves time by making it easy to type and edit requirements; provided reusable requirements; help in thinking how to go about writing the requirements; helped put forward ideas in formal manner; better if can be auto-filled (e.g., input field for resource, subject, action as these fields are same within a template);
- *More templates*: more template choices would be even helpful; good but not exhaustive;
- *Apply with caution*: not all requirements in the template may be relevant; should not be considered as an exhaustive list; might lead to choosing templates arbitrarily, without carefully thinking about security requirements;

Only one participant in our study had around five years of academic and work-related experience in security. Although this participant found some of the templates to be "not very intuitive", the participant applied templates successfully and performed the best in his/her task group (UC1, treatment) in terms of quality, quantity and efficiency. Another participant, who had one to two years of work-related experience in security cautioned that "Using the tool over time might discourage developers to think for themselves and assume the tool provides complete coverage."

We received the following feedback in terms of the use of requirements generated through the templates:

- *Help identify core set of security requirements*: helpful as a starting point for security requirements engineering; provides a solid list of initial security requirements;
- *Help identify correct and relevant requirements*: accurate and relevant to the description; for a generic template to provide such specific requirements was impressive;
- *Help in developing secure systems*: help in designing security measures earlier on; help in identifying potential threats; help think about security of data and users;

5. THREATS TO VALIDITY

We have considered the following threats to validity [19].

5.1 Internal Validity

Selection: In effect, the round-robin assignment approach randomly assigned participants to treatment and control groups, as well as to one of the two use cases. Unbalanced groups in terms of participant expertise in the given task could result. However, based on the background information of participants, groups were evenly balanced in terms of expertise.

Instrumentation: When measuring time spent on the task, we automatically recorded every time a participant saved or resumed the task. The recorded timestamps helped us assess the actual time spent on the task at a more granular level, compared to self-reporting by participants.

5.2 External Validity

Representativeness of sample population: Participants in the study were enrolled in a graduate course on software security. The study was conducted towards the end of the course, when participants had been exposed to concepts related to security principles, practices and tools. Participants can be considered representative of entry-level, non-expert security practitioners.

Task representativeness: Participants identified security requirements based on a single use case scenario. Additional context for the system and problem domain may help in considering additional requirements.

Experimental constraints that limit realism: Participants used a limited amount of time to complete the task which may affect the quality and completeness of identified security requirements.

5.3 Construct Validity

Hypothesis guessing: Participants were not aware of the existence of treatment versus control groups or whether they belonged to different groups. Participants were told only that they were supposed to perform the task of identifying security requirements from a given scenario. This single blinding was used to minimize biases towards viewing one requirements elicitation process favorably as compared to the other.

5.4 Conclusion Validity

Reliability of measures: As is often the case with experiments in software engineering, we could not employ double-blinding when reviewing the participants' responses. Determining whether a participant belonged to the treatment or control group was obvious, since participants in the treatment group used security requirements templates with standardized wording. In contrast, participants in the control group specified requirements in their own words. Care was taken to minimize biases during the evaluation of the responses by devising quantitative measures whenever possible, having multiple independent evaluators and using a standard oracle created beforehand.

6. LESSONS LEARNED

The task given to students was mandatory to complete for class participation, but participation in the study was not mandatory. Almost 93% of the students agreed to participate in the study. Students may be more willing to participate in a study when it is a relevant part of the educational experience.

To keep the responses anonymous, we distributed a separate sheet of paper to all students asking: *Briefly explain what you learned as part of this classroom exercise*. All students who submitted the sheet to the teaching staff received participation credit for the class exercise, irrespective of their participation in the study or of the quality of their responses. The lack of a detailed grade on the exercise might have affected their motivation to spend more time and effort in identifying high-quality security requirements. To ensure that participants diligently work on a task, they need further incentive than just a participatory grade. Either the experiment setting needs to be changed to give dedicated time during a lab or lecture to completely perform the task, or the participants need some sort of grades, scoring, or personalized feedback to place more focus on producing quality responses.

Further, we recommend that to ensure task completeness, participants either need a step-driven (i.e., wizard) approach to perform the task, or progress indicators to effectively identify remaining work. Such clues might enable participants to think more about the task and how much effort is expected of them.

One-third of the participants in the treatment group did not completely fill-in the templates to produce concretized security requirements. We plan to improve our tool by automatically filling-in the templates when applicable. We may also check that participants have provided missing information (such as resources and actions) before they can add the requirements suggested by the template to the final set of identified security requirements.

7. BACKGROUND AND RELATED WORK

Security Requirements Engineering (SRE) has gained focus in recent years with emphasis on identifying security requirements early in the software development lifecycle. Mellado et al. conducted a systematic review of SRE approaches [13] to summarize existing methodologies. Efforts to automate parts of the SRE process have resulted in an organizational learning approach to SRE [9] that identifies when a security requirement is added to a natural language artifact. This approach helps build a repository of security requirements for consideration and reuse in subsequent projects. In contrast, our process goes beyond identifying explicit security requirements in natural language artifacts. We identify implied security-relevant statements in natural language artifacts and associated security objectives. Furthermore, we provide templates to identify functional security requirements to satisfy the security objectives implied by statements in natural language artifacts.

Yskout et al., [20] conducted a controlled experiment to study the effect of annotating security patterns with additional details such as objectives, trade-offs, and relations among patterns in hardening of software architectures. Their findings indicate that annotations helped the participants to focus on applicable patterns and reduced the number of alternative patterns under consideration. However, the annotations did not reduce the time spent on carrying out the task. We also annotate statements in the project documents, such as requirements specifications or use case scenarios, with implied security objectives and then suggest applicable security requirements templates. Similar to findings by Yskout et al., [20] participants using our process did not spend less time on task but utilized the time more efficiently by identifying more applicable security requirements.

8. CONCLUSIONS AND FUTURE WORK

We conducted a controlled experiment to evaluate the use of automatically-suggested templates in identifying implicit security requirements as compared to a manual approach without the guidance of templates. We presented information about the security objectives implied by statements in the given use case and suggested security requirements templates to consider when identifying applicable security requirements. Participants in the treatment group identified significantly more security requirements in the same amount of time as the control group.

We did not observe any significant differences in the quality of the identified requirements by the treatment and control group. One-third of the participants in the treatment group did not fill-in the templates they selected during the task. The incomplete templates affected the overall quality of the participant's response.

Participants in the control group who did not use the reference material (example security requirements and security objectives) either specified too general requirements or too specific requirements, talking in terms of mechanisms. Participants in the control group, who used the example security requirements as reference, specified better quality requirements than those who did not use the reference material in the control group.

Our process helps participants identify a core set of relevant security requirements and focuses critical thinking around software security concerns. However, the requirements generated using our process should be considered as a starting point to build upon and not as an exhaustive list of security requirements. Our process assists requirements engineers to identify security requirements that may otherwise be overlooked.

In future, we plan to evaluate our process in real industrial settings. We also plan to compare our approach with other approaches, such as checklists, in addition to a control group.

9. ACKNOWLEDGMENTS

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 NCSU Realsearch group for their helpful comments on the paper.

10. 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] 2004. Standards for Security Categorization of Federal Information and Information Systems: http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf

[4] Boslaugh, S., 2012. The Pearson Correlation Coefficient. In *Statistics in a Nutshell* O'Reilly Media, Inc.

[5] D. Mellado, C. Blanco, L. E. Sánchez, and Fernández-Medina, E., 2010. A Systematic Review of Security Requirements Engineering. *Computer Standards and Interfaces 32*, 4 (Jun.), 13.

[6] Firesmith, D.G., 2003. Engineering Security Requirements. *J. Object Technology 2*, 1 (Jan-Feb.), 16.

[7] Jedlitschka, A., Ciolkowski, M., and Pfahl, D., 2008. Reporting experiments in software engineering In *Guide to Advanced Empirical Software Engineering* Springer London, 201-228.

[8] Kitchenham, B. and Charters, S., 2007. *Guidelines for performing systematic literature reviews in software engineering*. Technical Report EBSE-2007-01 School of Computer Science and Mathematics, Keele University.

[9] Kurt Schneider, Eric Knauss, Siv Houmb, Shareeful Islam, and Jürjens, J., 2012. Enhancing security requirements engineering by organizational learning. *Requirements Engineering 17*, 1, 35-56.

[10] Landis, J.R. and Koch, G.G., 1977. The measurement of observer agreement for categorical data. *Biometrics 33*, 1, 159-174.

[11] Lane, D.M., Research Design. In *Online Statistics Education: An Interactive Multimedia Course of Study*, Rice University.

[12] Linstone, H.A. and Turoff, M., 2002. The Delphi Method: Techniques and Applications. *Technometrics 18*, 3, 363.

[13] 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.

[14] Meneely, A., Smith, B., and Williams, L., 2012. Appendix B: iTrust electronic health care system case study. In *Software and Systems Traceability* Springer Verlag.

[15] Riaz, M., King, J., Slankas, J., and Williams, L., 2014. Hidden in Plain Sight: Automatically Identifying Security Requirements from Natural Language Artifacts. In *Proceedings of the 22nd International Conference on Requirements Engineering* (Karlskrona, Sweden, Aug 25-29 2014), IEEE.

[16] Salini, P. and Kanmani, S., 2012. Survey and Analysis on Security Requirements Engineering. *Computers and Electrical Engineering 38*, 13.

[17] 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.

[18] Viera, A.J. and Garrett, J.M., 2005. Understanding interobserver agreement: the kappa statistic. *Fam Med 37*, 5, 360-363.

[19] Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., and Wesslen, A., 2000. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Sweden.

[20] Yskout, K., Scandariato, R., and Joosen, W., 2012. Does organizing security patterns focus architectural choices? In *Proceedings of the International Conference on Software Engineering (ICSE '12)* (Zurich, Switzerland, 2-9 June 2012), 617-627.