Identifying emerging security concepts using Software artifacts through an experimental case

Gastón Márquez, Paulina Silva
Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile
{gmarquez,pasilva}@alumnos.inf.utfsm.cl

Rene Noël
Escuela de Ingeniería Civil Informática
Universidad de Valparaíso
Valparaíso, Chile
rene.noel@uv.cl

Santiago Matalonga
Departamento de Ingeniería de Software
Universidad ORT Uruguay
Montevideo, Uruguay
smatalonga@uni.ort.edu.uy

Hernán Astudillo
Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile
hernan@inf.utfsm.cl

Abstract—The development of secure software systems is an increasingly important research topic in software engineering. Several authors have proposed methods, techniques and tools to software development practices in order to identify and/or mitigate security threats. These methods and techniques are based in traditional software engineering artifacts, such as Use Cases, Activity Diagrams and Domain Models. However, the lack of scientific evidence of the quality or efficiency of these methods, leads us to question if this approach is necessary for software security experts. This article proposes an experimental approach to explore if software development artifacts are relevant when making security decisions in software development, and how are they used. We have designed a survey in order to ask these questions to software security and architecture experts. We used the Constant Comparison Method in order to find emerging security theories about software artifacts, grounded in the answers of the experts. Our results add experimental evidence into the use and usefulness of software development artifacts in helping to reduce security vulnerabilities in practice, from the experts’ point of view. Our results add experimental evidence into the use and usefulness of software development artifacts to evaluate the security from the point of view of the experts. Our evidence suggests that not all software artifacts are equally useful in the design of secure software, considering the "Use Cases" and "Class Diagrams" as the most useful artifacts according to our respondents. Also, our evidence suggest that experts do not agree in the importance of analyzing security concerns through the whole software life cycle, nor in the abstraction level required for this task.

I. INTRODUCTION

Software Architecture is a set of significant decisions about the organization of a software system [15]. These decisions allow controlling the software quality attributes. One of the most critical quality attributes is security, due to its increasing importance and huge impact in the overall quality of the system. An approach to handle security design decisions through knowledge reuse are architectural tactics and security patterns. Architectural tactics are introduced by Bass et al. [3]. The authors propose the use of tactics in order to address availability, modifiability, performance, security, testability and usability security attributes, and define them as “a design decision that influences the control of a quality attribute response”.

The original definition of each software tactic is a high level description of a design decision, but [13] and [25] propose a formal approach for their specification and validation. Security patterns define solutions to handle threats or to fix a vulnerability [5]. Security Patterns include a solution to a security problem and several sections that define their use, applicability, advantages, and disadvantages.

Both approaches differ in the detail of the design decision and in the guidance to its implementation. While tactics are high level definitions and not specify the use or modification of software artifacts, security patterns integrate with the whole software development life cycle and artifacts such as Class, Component and Sequence UML diagrams. This difference guide us to question if, in practice, software artifacts are useful for security experts in order to make security decisions.

This article presents the first results of an experimental approach to explore the use and usefulness of software artifacts for making security design decisions, from the point of view of security experts. To extract information from experts, we performed a survey with questions related to the use of artifacts of a software project to make security design decisions. We designed the survey with two goals: 1) Gather direct answers about the use of a set of software artifacts (i.e., given a Use Case Diagram, we asked a yes/no question: “is this artifact useful for making security decisions?”) and 2) Explore key concepts about what do the experts look for in this artifacts. (i.e., given the same Use Case diagram, we asked an open question like “how does this artifact help you to make security design decisions?”). For this second goal, we used Grounded Theory (GT) [4] to build emerging concepts representing the information that the experts need to make security decisions.

The structure of the article is composed of the section II where we describe the state of art, focused in the use of software artifacts in security techniques with scientific evidence of its applicability and impact. In section III, we present key concepts related to software artifacts and the research methodology. Section IV presents details of the planning and execution of the survey, and in section V, we discuss the results of the experiments and future work guidelines.

II. SYSTEMATIC MAPPING

Through the years, several methods, techniques and tools for building secure software systems have been proposed, but only a few of them have reported scientific evidence of its applicability and effect over software quality. We performed
a systematic mapping of literature following the procedures described by Kitchenham et al. [14] to extract information about the use of software artifacts in methods, techniques and tools that report scientific evidence of its use and/or effect. The protocol followed for this review can be found here. The following table present our main findings: first, we describe the works and their authors, and then in the Table 1 we describe a summary of the proposal of each work. Column A shows the work number (from the next list), column B specifies if the article presents quantitative or experimental evidence, column C identifies if the proposal is a technique for identify security treats or to mitigate them, and column D details if the proposal uses software artifacts.

1) Xu et al. [26], “Threat-Driven Modeling and Verification of Secure Software Using Aspect-Oriented Petri Nets”  
2) Georg et al. [6], “Verification and Trade-Off Analysis of Security Properties in UML System Models”  
3) Toval et al. [23], “Requirements Reuse for Improving Information Systems Security: A Practitioner’s Approach”  
4) Irvine et al. [9], “An Approach to Security Requirements Engineering for a High Assurance System”  
5) Sindre et al. [21], “Eliciting security requirements with misuse cases”  
6) Suleiman et al. [22], “Evaluating the effectiveness of the quality requirements engineering (SQUARE) method: a case study using smart grid advanced metering infrastructure”  
7) Kalloniatis et al. [11], “Evaluating cloud deployment scenarios based on security and privacy requirements”  
8) Sanchez et al. [20], “ModelSec: A Generative Architecture for Model-Driven Security”  
9) Matulevicus et al [16], “Syntactic and Semantic Extensions to Secure Tropos to Support Security Risk Management”  
10) Mouratidis et al. [17], “A framework to support selection of cloud providers based on security and privacy requirements”  
11) Yu et al. [27], “Automated analysis of security requirements through risk-based argumentation”  
12) Opdahl et al. [18], “Experimental comparison of attack trees and misuse cases for security threat identification”  
13) Truillo et al. [24], “An engineering process for developing Secure Data Warehouses”  
14) Karpati et al. [12], “Comparing attack trees and misuse cases in an industrial setting”  
15) Razzaq et al. [19], “Ontology for attack detection: An intelligent approach to web application security”  

As Table 1 suggests (column B) there is limited experimental evidence on the application of tactics and patterns to the problem of designing security. The reader can evaluate that only 3 of 15 are experimental comparison, most of the selected literature is composed of case study research.

In any case, we claim that there is insufficient evidence to answer our research question of whether software architects make use of software artifacts when designing security techniques.

### III. RELATED CONCEPTS

A software development artifact describes information about a software product. Examples of the software artifacts are Use Cases, Software Architecture, Object Collaborations and Class descriptions. Software development artifacts can be very abstract, such as the vision from the software system, or very concrete, such as the source code [8].

A Grounded Theory (GT) is a group of integrated conceptual hypotheses generated to produce a theory about a substantive area [7] which can be interpreted as a method for generating theory from data (Figure 1).

Researchers performing GT, do not start with preconceived notions about the phenomenon under study, but rather they begin with an area of study and then what is relevant is allowed to emerge from the research. The Constant Comparative Method (CCM) is a process in which any collected data is compared with previous data that was collected in one or more earlier studies [7] (Figure 2).

---

Table I

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>Systematic case study</td>
<td>Modeling and mitigating security</td>
<td>Petri Nets to model security threats</td>
</tr>
<tr>
<td>2)</td>
<td>Systematic case study</td>
<td>Identification and evaluation of security threats</td>
<td>Functional models</td>
</tr>
<tr>
<td>3)</td>
<td>Systematic case study</td>
<td>System security with Requirement Engineering</td>
<td>Not apply</td>
</tr>
<tr>
<td>4)</td>
<td>Systematic case study</td>
<td>Method for security requirement elicitation</td>
<td>Not apply</td>
</tr>
<tr>
<td>5)</td>
<td>References in research projects</td>
<td>Method for identification of security threats</td>
<td>Use Cases and Misuse Case</td>
</tr>
<tr>
<td>6)</td>
<td>Case study</td>
<td>Method for security requirements elicitation</td>
<td>Architecture diagram</td>
</tr>
<tr>
<td>7)</td>
<td>Real case study</td>
<td>Method for security requirements elicitation</td>
<td>Modeling language (i* language)</td>
</tr>
<tr>
<td>8)</td>
<td>Method in example</td>
<td>Security requirements management method</td>
<td>MDS specification</td>
</tr>
<tr>
<td>9)</td>
<td>Case study</td>
<td>Rika management modeling method</td>
<td>Secure Tropos language</td>
</tr>
<tr>
<td>10)</td>
<td>Case study</td>
<td>Security requirement method analysis (cloud computing)</td>
<td>Not apply</td>
</tr>
<tr>
<td>11)</td>
<td>Case study</td>
<td>Security threats identification method (rebuttals)</td>
<td>No apply</td>
</tr>
<tr>
<td>12)</td>
<td>Experimental comparisson</td>
<td>Comparisson security between two threats identification</td>
<td>UML as misuse case and attack tree</td>
</tr>
<tr>
<td>13)</td>
<td>Case study</td>
<td>A method for design secure warehouses</td>
<td>Tropos framework adapted</td>
</tr>
<tr>
<td>14)</td>
<td>Experimental comparisson</td>
<td>Comparisson security between two threats identification</td>
<td>Misuse case and attack tree</td>
</tr>
<tr>
<td>15)</td>
<td>Eval. of quality metrics of ontology</td>
<td>Survey</td>
<td>OWL-DL (Descriptive Logic Based Web Ontology Language)</td>
</tr>
</tbody>
</table>

---

1https://goo.gl/kwAOtL
This is a continuous ongoing procedure, because the theories are formed, enhanced, confirmed, or even discounted as a result of any new data that emerges from the study. As we know, one way in which data can be constantly compared throughout a research study is by means of coding. Although usually the data that we code arises from sections of text that we have transcribed from interviews, it can also be excerpts from audio or video recordings, or images. All these are similarly coded to ensure consistency of coding within any particular research study.

IV. SURVEY DESCRIPTION

In this section we will describe the guidelines that we used to execute our experiment. An adapted approach described by Jedlitschka et al. [10] for reporting experiments in software engineering was used in the following sections.

A. Background

The subjects of the survey are security experts, that will be asked to answer questions related to a software project and its artifacts. The software project embraces the development of an information system that allows the description of Educational Domains through software ontologies. This kind of description is a labor that requires several man hours and is complex to be approached by one expert or by a limited experts team, because the quality of the final descriptions are limited by the knowledge, work spaces and time of the team work.

As a solution to this problem, the software platform will allow experts the distributed description of domains. The main goal is to reduce the effort and resources for this work, through the standardization of the work methodology and the use of trust schemes to create consensus or acceptation of descriptions, i.e. using feedback and algorithms to determine the final acceptation. To develop this platform, several artifacts and documents have been developed. We selected this project considering the detailed documentation and the lack of security artifacts, documented decisions or any other explicit references to security concerns. In the survey we presented the following artifacts: Use Case Diagram (UCD), Conceptual Model (CM), System Architecture Diagram (SAD), Class Diagram (CD) and Database Internal Logical Model (DILM).

B. Survey Planning

In this subsection we will explain the planning of our experiment with objective to validate our results. The next topics that it will be approached in this section are: Goals, Survey Material and Tasks.

1) Goals: Here we will define in more concrete terms the principal manipulations of the experiment. Next, the goals will be described:

- Goal 1: Identify the corresponding artifacts of the software engineering that allow to experts evaluate the security in practice.
- Goal 2: Capture emerging security concepts since the expert’s point of view when they evaluate vulnerabilities in software engineering.

2) Survey material: To execute our survey, the research team created 12 questions through a process of several iterations with experts in the security area, to refine questions contents (software artifacts and description detail), the strategy for ordering the questions (we present a general question first, then specific questions for each artifact, and then another general question), and estimated execution time of each question.

Next, we will describe the questions for the survey but is necessary to note that the questions $Q_3, Q_5, Q_7, Q_9$ and $Q_{11}$ are responses such as Yes/No, because when questions iterations were performed to obtain the final list of survey questions, the experts recommended us to apply questions from a general level to a particular, this means that for every artifact is recommended check first if the artifact allows evaluate the security, answering yes or no. If the expert considers that the artifact allows evaluate the security, other questions were made with a greater degree of detail (questions $Q_4, Q_6, Q_8, Q_{10}, Q_{12}$):

- $Q_1$: What documents, models or artifacts of analysis of software design you consider to take security decisions?
- $Q_2$: Since your answer of the above question, what do you see to evaluate the security in the document, model or artifact?
- $Q_3$: Based in the Use Case Diagram and the actor description, you consider if this artifact allows evaluate the security in the systems?
• Q4: If so, what is your argument to determine that the system has vulnerabilities?
• Q5: Based in the Conceptual Model of the system, you consider that this artifact allows evaluate the security in the systems?
• Q6: How do you evaluate what element of the Conceptual Model is susceptible to security threats?
• Q7: Since the Architecture System Diagram, do you think that this artifact can evaluate the security?
• Q8: If so, what do you see to establish if this diagram is secure or not?
• Q9: Based in the Class Diagram, do you think this artifact help you to evaluate security of the system?
• Q10: If so, what do you look to evaluate the security?
• Q11: The Database Internal Logical Model, is useful to evaluate the security?
• Q12: If so, what aspect you review to determine if the database has vulnerabilities?

3) Tasks: Subjects are asked to read the short description of the system, which is allocated inside of the survey, and then answer the questions. For a deeper analysis, subjects can access to the original document where is described all details of web platform in study.

C. Execution

In the execution of our experiment, we used the tools offered by the platform SurveyMonkey [1]. We selected the sample by convenience; nonetheless, the respondents come for different backgrounds and two different latin american countries, which, in spite of limited generalization due to convenince, it adds the capacity to generalize the results. The resume of the experts considered in the sample are described in Table II.

<table>
<thead>
<tr>
<th>Expert</th>
<th>Country</th>
<th>Exp. (years)</th>
<th>Area of interest</th>
<th>Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert 1</td>
<td>Chile</td>
<td>20+</td>
<td>Software Quality</td>
<td>Academic</td>
</tr>
<tr>
<td>Expert 2</td>
<td>Chile</td>
<td>20+</td>
<td>Software Security</td>
<td>Academic</td>
</tr>
<tr>
<td>Expert 3</td>
<td>Chile</td>
<td>5+</td>
<td>Software Projects</td>
<td>IT Architect</td>
</tr>
<tr>
<td>Expert 4</td>
<td>Chile</td>
<td>4+</td>
<td>Software Projects</td>
<td>IT Architect</td>
</tr>
<tr>
<td>Expert 5</td>
<td>Uruguay</td>
<td>7+</td>
<td>Software System Infrastructure</td>
<td>System Architect</td>
</tr>
<tr>
<td>Expert 6</td>
<td>Uruguay</td>
<td>15+</td>
<td>Software Projects</td>
<td>IT Architect</td>
</tr>
<tr>
<td>Expert 7</td>
<td>Uruguay</td>
<td>11+</td>
<td>Software Projects</td>
<td>IT Architect</td>
</tr>
</tbody>
</table>

Respondents answered the survey un attended via SurveyMonkey user interface. Once all answers had been collected, we proceeded to the data analysis section.

D. Results and analysis

Once compiled the answers of the survey, our approach is aimed to work with the proposed goals in our investigation. Identify what is the vision of the expert or how they handle the concept of security thought of they own experience, allows interesting challenges related to create theories from the practice.

So, the analysis using CCM we adapted to obtain an comparison table with three variables: concept, indicators and emerging concepts. The data of this table has been compiled and refined by two of our researchers of this project with the objective to cross the different analysis and get a more common vision of the concepts, indicators and emerging concepts. Next, we will show in the Figure 3 the results of the Yes/No questions and in the Table III the summary of the CCM applied in the survey.

![Figure 3. Chart graph analysis of the questions Q3,Q5,Q7,Q9 and Q11](image)

As we can see, the comparison table has a summary of the concepts and indicators extracted from the answers of the experts. Also, the table contains the emerging concepts that our team could create.

From the results, we have observed experts analyze and perceive security concerns at different levels of abstraction. In Q1 the aim was to analyze what the experts use to evaluate security when artifacts are used to describe software development. Among the responses obtained, documentation and specification requirements are identified as useful artifacts to assess security threats in software development. Citing one of the answers: "...Architecture Document Information, Security Requirements, Scope Documentation (knowing what the system is), Requirements (or goals), Security policies of the organization (if any)..." allow us to verify that the extraction of the emerging concept for Q1 is connected with software abstractions (artifacts) that shall represent security decisions. However, other responses indicate that to some experts, there is no need for security details in the technical documentation or in the requirements specification. A divergent answer, and also an interesting point of view, is that security should be discussed with external dynamic variables that can affect
Table III

<table>
<thead>
<tr>
<th>Concept</th>
<th>Indicators</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Security decision</td>
<td>To interpret security, it is necessary to abstract the concept</td>
</tr>
<tr>
<td>Q2</td>
<td>Security Predisposition</td>
<td>To evaluate security, must exist a conceptualization of what is security for the experts</td>
</tr>
<tr>
<td>Q3-Q4</td>
<td>Identifying security vulnerabilities in Use Case Diagrams (UCD)</td>
<td>Security specifications can be expressed in UML UCD</td>
</tr>
<tr>
<td>Q5-Q6</td>
<td>The elements of the conceptual model are susceptible to security threats</td>
<td>Respondents did not give open answers</td>
</tr>
<tr>
<td>Q7-Q8</td>
<td>The architectural diagram allows to evaluate security</td>
<td>Some kind of Software Architecture diagrams can help to evaluate vulnerabilities</td>
</tr>
<tr>
<td>Q9-Q10</td>
<td>The Class Diagram identifies security vulnerabilities</td>
<td>Security vulnerabilities can be inferred in the UML Class Diagram</td>
</tr>
<tr>
<td>Q11-Q12</td>
<td>The internal logical model of the database can identify security threats</td>
<td>Respondents did not give open answers</td>
</tr>
</tbody>
</table>

software development, and not in static documentation. The context of the question Q1 was to initiate a global survey question to see the views of the experts, then to go into more detail in Q2 wondering what they look when evaluating the security artifacts that were mentioned in Q1. This question allowed us to observe with greater detail elements that are used in practice when assessing security. As an example of the responses “...In the case of architecture documentation, deployment and implementation specifications are useful to identify if there are private applications exposed to the outside (Internet), and are not vulnerable to major risks cited in the latest TOP 10 OWASP, in particular prevent external applications to query SQL directly and instead invoke Web Services. And considering infrastructure document, there is always a firewall with IPS as first barrier from the Internet, and then a Web Application Firewall. Both besides protecting, they leave a valuable trace of what happened to the protocols set forth...” for validating our emerging concept for Q2, show us the necessity of conceptualize security decisions. For most experts, security abstraction is not clear because we have realized that there is a difference in assessing the security device and runtime level. Specifically on the question Q2, experts point to verify level security technology used for software development, for example, the vulnerabilities described in OWASP (Open Web Application Security Project) should be considered in the final stage development when testing the final product. But other experts detailed that security should be treated initially as a quality requirement. Therefore, this makes us think that the conceptualization of security has not been well defined in the community. The aim of the other questions we have made is whether the most common artifacts that are created, in the developing software projects, are useful to assess security. Our study determined that from the diagrams described in section 4.1, the diagrams that experts uses to assess security are UCD and CD. Among the responses we collected it is mentioned that: "...In use cases, it is possible to identify misuse cases. Example: Load File should take the case of misuse (or abuse) load malicious file. For use Approve Transaction, the case of misuse of Operation Approval by unauthorized person. In short, can identify high-level test to be performed on the system and define them by risk (the most risky are made, the less risky if given the time)...", on the side of UCD and for the CD side, the experts mention that "...May give some clue to detect threats for classes, for example SQL injections in MySQLFactory, free and classes that inherit from them...". This allows us to deduce at first instance that for some experts, the domain problem is not mentioned in the discussion to identify vulnerabilities, but artifacts that describe the interactions the user/system level could identify possible causes of security risks. The concepts that we get to the questions associated with the artifacts, they approach that we can specify security in artifacts.

To summarize, CCM applied using GT gives us the ability to view theories from practice and in our case, in the first iteration emerging concepts allow us to establish initial theories regarding identify security vulnerabilities generated in artifacts software development. In order to define a more complete theory using CCM, we have to identify other edges of the emerging concepts, such as possible properties.

V. CONCLUSIONS AND FUTURE WORK

Our survey allow us to have a first impression of the practical use of software artifacts for making security design decisions. For some of the respondents, security should be addressed on a project runtime level, embracing security concerns during the implementation activities. For them, technological properties provided by development frameworks and programming languages can solve most of security issues. In other words, security is a requirement that must be handled in the implementation/execution phase. But, for other experts, security can be analyzed throughout the software development lifecycle, using software development artifacts. Questions Q7 and Q8 were especially meaningful, the experts agreed that the proposed architectural diagrams were not useful for making design decisions, but all of them declared that it was for the lack of information and detail of the presented diagrams, giving us the hint that they really need this artifact. Consistently, in responses to the question Q8,
experts mentioned that there are elements that can be evaluated in an architectural diagram such as layers, isolation between layers, display elements, among others. In spite of our limited sample, main emerging concepts associated with Goal 2 of our study guide us to draw the following conclusions: 1) There is no a common view between the experts in the level of abstraction required to analyze security concerns, 2) There is no a common agreement in the importance of making security decisions though the complete software development life cycle and 3) There is no a total agreement in the artifacts needed to analyze security concerns, although there is a shared view that static information views (domain models and database logical model) are not useful for this task.

Finally, the main contributions of the first phase of our experimental work could be summarized in the following points:

- We have found concrete answers for the use of software artifacts in security design decisions from a sample of experts from two Latin American countries, with different backgrounds and experience levels.
- We have raised three conclusions from their different views: there exist different levels of abstraction and conceptualization of security, there are different visions about the need of embracing security through all the software lifecycle, and data and domain models might not be useful to analyze security concerns.
- We also apply Grounded Theory, a novel research approach in software engineering.

As future work, we want to improve our application of the CCM method using more analysis phases, with the aim of creating categories and properties of emerging concepts in order to create a more complete emerging security concepts that are extracted from the experience of experts. We also want to improve our survey with more precise questions to obtain key facts that experts used to assess safety, and extend our sample of subjects to allow for greater generalization of the results.

VI. ACKNOWLEDGEMENTS

This work has been financially supported by Comisión Nacional de Investigación Científica y Tecnológica, FONDECYT regular 1140408.

REFERENCES


