Software development initiatives to identify and mitigate security threats
Silva, Paulina; Noël, René; Gallego, Mauricio; Matalonga, Santiago; Astudillo, Hernán

Published: 27/04/2016

Document Version
Publisher's PDF, also known as Version of record

Link to publication on the UWS Academic Portal

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the UWS Academic Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

This is an Open Access item distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited, and is not altered, transformed, or built upon in any way.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Methodologies to Identify and Mitigate Security Threats in Software Development

Paulina Silva¹, Rene Noel², Mauricio Gallego³, Santiago Matalonga⁴, and Hernán Astudillo¹

¹ Universidad Técnica Federico Santa María
{pasilva}@alumnos.inf.utfsm.cl
{hernan}@inf.utfsm.cl

² Universidad de Valparaíso, Chile
{rene.noel}@uv.cl

³ Universidad de Quindío, Colombia
{mauriciogallego81}@gmail.com

⁴ Universidad ORT Uruguay, Uruguay
{smatalonga}@uni.ort.edu.uy

Abstract. The effective building of secure software systems has been addressed by security experts and software development experts through several techniques for identify and mitigate security threats. Many techniques had been compiled, however, for most of these proposals there is few empirical evidence of its application in building secure software systems. A systematic mapping has been conducted to cover the existent technologies for identification and mitigation of security threats. A total of 10 different techniques covering threats identification and 8 covering the mitigation of threats were found. All the initiatives were integrated to at least one activity of the Software Development Lifecycle (SDLC), while 7 show signs of being adopted in the industry. The mapping found only 15 studies that covered 11 different initiatives. Only two techniques presented scientific evidence of its results through controlled experiments, while others selected studies presented informal case studies or examples.

Keywords: Security Patterns, Architectural Tactics, Security Threats, Secure Software Development, Systematic Mapping

1 Introduction

Security is a main interest topic in Information Technologies nowadays, and is considered a Quality Attribute of a System by software engineering (ISO 250010). From this point of view, several initiatives, like techniques, models and tools, have been proposed in order to add architectural and design decisions through the Software Development Life Cycle (SDLC) [31]. Some of these initiatives aim to help software engineers in the identification of security threats such attacks trees [26], the elicitation of security requirements with misuse cases [27], and the definition of security goals as in Secure Tropos [15], while others allow software engineers to mitigate these security threats or satisfy security requirements, by reusing security design decisions [6, 1], or supporting the decision making process [33]. However, the impact of these techniques, methods and models is hard to assess, considering the nature of software development process, the lack of standards to measure software security from the point of view of the software developer, and the difficulty of defining security requirements in an effective, clearly and unambiguous way [10]. Also, the application of these proposals could be hard to replicate in real contexts,
due to the dependence on the software process activities, artefacts and practices required as preconditions, that may vary between software development organizations or even between projects in the same organization. This paper presents a systematic mapping [22] to characterize techniques, methods, tools and processes proposed in Security and Software Engineering Journals, which at least show an example of the application of these proposals. The goal of the study is to identify these proposals, characterize them by level of empirical evidence of its application, identify their use of software artefacts or notations, and identify the stage of the SDLC that these proposals support. The rest of the paper goes as follows: Section 2 presents the related work, reporting previous work of the authors in software security design studies, relevant systematic mapping publications that support our approach to answer our research questions, and some proposals of techniques, methods and tools that initially guided our search. Section 3 presents the design of the systematic mapping, while section 4 presents details of the execution and results. The validaty of our work will be review in section 5. Finally, section 6 presents main conclusions and high level answers to our research questions.

2 Related Work

Authors have conducted several studies aiming to explore software design decisions for developing secure software systems. An experimental approach was initially taken to compare software security tactics and security design patterns [17], and several questions arose when trying to operationalize the identification and mitigation of security threats. These questions guided the authors to search for some security threat identification methods with demonstrated efficiency to be added to the experimental process, as a previous step to apply security tactics or patterns. In a first search, no methods with scientific evidence of its impact were found, which motivated the present systematic mapping. Authors choose a well explained methodology for a first experimental approach to threats identification: Misuse Activities [3]. Results of applying this identification method can be found in [18], which show statistically significant differences when performing a guided identification process versus an ad-hoc approach. Also, authors have on going research initiatives to explore the importance of software artefacts over security decisions [14], and the structure of the Architectural Tactics for security decisions [7]. Whether a number of methodologies exist as mentioned in [31], the study [32] in which the security expert’s opinion about the State-of-Practice of initiatives in the software development process was analized, revealed that a lot of them do not surpass the State-of-Art even if they are considered good practices. The study establishes requirements for a methodology for building secure systems, including that proposals must consider the compatibility of the technique with the commercial environment of developers and their current skills.

A software engineering systematic map (SM) is a defined method to build a classification scheme and structure a software engineering field of interest. When working with Systematic Mappings, it is difficult not to relate them to Systematic Reviews. The main difference between them lies, according to [22], relays in the definition of goals, breadth, validity issues and implications of the studies. In [25] a series of rigorous guidelines to make a systematic mapping are presented, this directives constitute a well defined method for the revision of the mapping, with the potential to reach close to a high-quality systematic review. An example of the application of SM to answer research questions similar to the motivations of this study is presented in [5]: a SM is done with the goal of finding information about
usability evaluation methods, which have been used by researchers to evaluate web applications and their relationship with the web development process. Main results are the identification of most reported methods, and research gaps aiming to the validations of these methods. The study in [9] presents a work in which a systematic revision is done with the goal of finding a trend in the definition of quality in the Model Driven Engineering. Main contribution of the study is the identification of a variety of multiple quality interpretations as consequence of the diversity in MDE compliance works.

Previous initiatives for documenting existing security techniques, methods, tools and methodologies can be found in [31]. Also, a field study to identify requirements for a methodology for building secure systems can be found in [32].

3 Methodology/Research Design

The aim of this investigation is to cover the existent initiatives and experience reports seen as methodologies, methods, techniques and tools, for identification and mitigation of security threats during Software Development in order to build Secure Systems. An “initiative” is a method, technique, proposal or methodology, among others. No comparison or evaluation is done between the studies, so we could classify this work as a Systematic Mapping study (SM). In a SM we do not consider specific outcomes or experimental designs in our study, since we need a broad overview of the research area as a whole; this is also why this study is not a systematic review. In order to design and operate the study, we followed the guidelines proposed by Petersen [22], and the documented the protocol following Biolchini et al. definitions [2].

3.1 Research Questions

As commented in the related work section, no systematic mappings or reviews were found when searching for technologies for security threats identification and mitigation. Based on this main objective, we elaborated the following research questions:

1. What are the existing initiatives or methods of identifying security threats and attacks?
2. What threat identification methods are related to Security Tactics?
3. What other Mitigation Methodologies exist when developing Software, to conceive Secure Systems?

The first question is the Main research question, while the second and third are Secondary research question.

3.2 Identification of Population, Intervention and Amplitude

We followed P.I.C.O. [21] structure to define the search string, but with the considerations presented in [22], which suggests only use Population and Intervention.

- Population: Software Development or Design related work
  Keywords: “software”, “design”, “engineering”.
- Intervention Control: Security threat identification or mitigation methodologies/initiatives/guidelines
  Keywords: “security”, “threat”, “attack”, “identificate”, “mitigate”, “minimize”, “elicitate”, “elicitation”.
  Keywords: “model”, “guideline”, “checklist”, “method” and “methodology”.
  Keywords: “model”, “guideline”, “checklist”, “method” and “methodology”.
We excluded some words:
- “System”: It is a word that encompasses a lot of research areas.
- Use of “Service” or “System” for population: returns results from other areas, which are of no use to this work.
- Plural words were not used in the search string.
- Complete words were not used, we use the prefixes of them.

### 3.3 Construction of a Search String

The initial keywords were found out by asking our experts and using the control studies we defined in section 3.2; synonyms were added to the keywords already found. The keywords that represent the core of the research are: “Threat”, “Attack”, “Identify” and “Mitigate”. As we mentioned in section 3.2, we used the prefixes of the word as a consequence of our search procedure’s definition. This is further discussed in section 3.6.

<table>
<thead>
<tr>
<th>Term</th>
<th>Keywords obtained and synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>(“software” OR “design” OR “engineer” OR “develop”)</td>
</tr>
<tr>
<td></td>
<td>(“securi” OR “privacy” OR “integrity” OR “confidential” OR “available” OR “accountable”) AND (“threat” OR “risk” OR “attack” OR “requirement” OR “vulnerable”) AND (“identif” OR “mitig” OR “minimiz” OR “elicit” OR “enumer” OR “review” OR “assur”)</td>
</tr>
<tr>
<td>I</td>
<td>OR “elicit” OR “enumer” OR “review” OR “assur”)</td>
</tr>
<tr>
<td></td>
<td>AND (“model” OR “metric” OR “guideline” OR “checklist” OR “template” OR “approach” OR “strateg” OR “method” OR “methodolog” OR “tool” OR “technique” OR “heuristic”)</td>
</tr>
<tr>
<td>String</td>
<td>(P AND I)</td>
</tr>
</tbody>
</table>

Table 1: Keywords used in the Search String

### 3.4 Source Selection

Works should be available on the web, through open access or subscription based journals (those we could get access to). The language selected for the studies is English. The list of journals was proposed and reviewed by four researchers from different domains (software engineering, software security, distributed systems and software architecture), two of them not related to the research team. The Source Identification List is shown in Table 2.

### 3.5 Inclusion/Exclusion Criteria

The Study Type selected consist of articles related to Secure Software Development, and articles selected by the Inclusion Criteria (Table 3). Studies with at least one Exclusion criteria were not selected.

Two readers evaluated each study. We used the following acceptance criteria in case the evaluators do not agree which studies should be accepted.
- If the two readers accept: The study is included.
- If One reader accepts and one is in doubt: The study will be discussed in group.
- If the two readers excluded the study: The study is not included
- All conflicts were reviewed by a third researcher.

While the two readers are evaluating, they should document the rationale for excluding the article.
Table 2: Source Selection

<table>
<thead>
<tr>
<th>Journal Name</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Journal Information and Software Technology</td>
<td><a href="http://www.journals.elsevier.com/information-and-software-technology/">http://www.journals.elsevier.com/information-and-software-technology/</a></td>
</tr>
<tr>
<td>Journal of Universal Computer Science</td>
<td><a href="http://www.jucs.org/">http://www.jucs.org/</a></td>
</tr>
<tr>
<td>Requirements Engineering</td>
<td><a href="http://link.springer.com/journal/766">http://link.springer.com/journal/766</a></td>
</tr>
<tr>
<td>IEEE Transactions on Dependable and Secure Computing</td>
<td><a href="http://www.computer.org/web/tdsc">http://www.computer.org/web/tdsc</a></td>
</tr>
<tr>
<td>International Journal of Secure Software Engineer</td>
<td><a href="http://www.igi-global.com/journal/international-journal-secure-software-engineering/1159">http://www.igi-global.com/journal/international-journal-secure-software-engineering/1159</a></td>
</tr>
</tbody>
</table>

3.6 Search Strategy/Procedure

Our search strategy was to explore software development and software security journals, in order to cover studies reporting long term research in initiatives and accumulated scientific evidence.

1. We searched through studies from digital Journals. We selected the following information: Volume, Issue, URL (for the Journal-Volume), Title and Abstract.
2. All this information was saved in an Excel file, one file for each Journal revised.
3. From the Excel sheet where the studies are, we used a filter created by a Macro made in Visual Basic.
4. First Filter: This was done by searching with the Search String within the Abstract, and selecting those that had those terms in it. The Search String was created with the terms identifying the Population and Intervention. We manually evaluated the strength of the filter with one of the Journals used (IJSSE).
5. Second Filter: By reading again the title and abstract, we filter against the Inclusion and Exclusion Criteria (Table 3).
6. If any irrelevant or duplicate study appear, they should be removed.
7. Third Filter: From the second filter, we did a quick scan of the articles and tried to categorize them in the following cases: empirical evidence/experimentation, case study, proposal or systematic review. In this high level revision of the accepted papers we viewed Abstract, Introduction, Development of the work, Data obtained, Conclusions and other relevant sections of the paper; we considered that the abstract was not enough to judge a paper.
8. Fourth Filter: This is done by reading the complete text of the studies that were accepted by the third filtering. The fourth filtering is executed according
Inclusion Criteria | Exclusion Criteria
--- | ---
The paper mentions the Quality Attribute “Security” in some part of the work | Non-related Security article. Methodological proposals where security is not specified
Empirical or Experimental studies or Proposals (Methodological Definitions, Applied Case Studies or Experimental Application of Identification and Mitigation).
- Is there a concern to formalize? No matter if it was successful or not.
- Informal technical reports are useful (if it was wrote by someone of field). | Non Empirical or Experimental studies. Complete proceedings or books without specifying the articles. Exclude Thesis if is not the published version.
Applicable techniques during Software Development/Conception | Techniques used when a System is already deployed. Techniques related to certificate or security modeling or risk assessment. Articles not related to development or assurance of systems, for example, curriculum proposals or Security Quality Assurance
Discusses about strategies | No mention of strategies
Discusses about methods | No mention of methods

Table 3: Inclusion/Exclusion Criteria
to the Inclusion and Exclusion Criteria (Table 3). In this step, we tried to take some information that could help to speed up the Data Extraction.

4 Conducting the Systematic Mapping

The trial was done on 27-07-15 according to all the previous sections, where it was found out the following results:

<table>
<thead>
<tr>
<th>Journal</th>
<th>Number of Articles Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Journal Computers &amp; Security</td>
<td>29</td>
</tr>
<tr>
<td>Journal Information and Software Technology</td>
<td>19</td>
</tr>
<tr>
<td>Journal of Systems and Software</td>
<td>14</td>
</tr>
<tr>
<td>Journal of Universal Computer Science</td>
<td>6</td>
</tr>
<tr>
<td>Journal of System Architecture</td>
<td>6</td>
</tr>
<tr>
<td>Requirements Engineering</td>
<td>12</td>
</tr>
<tr>
<td>IEEE Transaction on Software Engineering</td>
<td>8</td>
</tr>
<tr>
<td>IEEE Transactions on Dependable and Secure Computing</td>
<td>10</td>
</tr>
<tr>
<td>International Journal of Secure Software Engineering</td>
<td>14</td>
</tr>
<tr>
<td>Information Security Technical Report</td>
<td>6</td>
</tr>
<tr>
<td>Journal of Information Security and Applications</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>127</strong></td>
</tr>
</tbody>
</table>

Table 4: Selected papers from first filtering (27-07-15)

To obtain the results on Table 4, we did the first phase filtering as its explained in section 3.6. No duplicate articles were found. Table 5 summarizes the quantity of articles selected in each phase. In the first phase filtering, the identification of the papers was performed in each Excel of the Journal selected, in total 127 papers
were identified by filtering its abstract with the Search String. In the second phase, 41 papers were filtered by the inclusion/exclusion criteria, and 0 were duplicated. After discussing between the revisors, we decided to add 1 more article, so a total of 42 papers went to the third phase filtering. The third filter was done by scanning 15 studies, which remained in the list after fully reading them and extracting the necessary information.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Number of Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Filter: In Abstract by Search String</td>
<td>127</td>
</tr>
<tr>
<td>Second Filter: Articles selected by inclusion criteria</td>
<td>41</td>
</tr>
<tr>
<td>After discussion between revisors</td>
<td>42</td>
</tr>
<tr>
<td>Duplicates</td>
<td>0</td>
</tr>
<tr>
<td>Third Filter: Lightly reading to other sections of the paper</td>
<td>15</td>
</tr>
<tr>
<td>Four Filter: Articles kept after full reading and Extraction</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 5: Resume of number of papers selected in each phase

4.1 Data Extraction

The 15 papers selected by our approach are described briefly in Table 6.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Initiative name</th>
<th>Initiative resume</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4]</td>
<td>CAIRIS</td>
<td>Create security models that are automatically generated from specified, declarative data rather than via direct manipulation.</td>
</tr>
<tr>
<td>[33]</td>
<td>Threat-Driven Modelling of Secure Software</td>
<td>Use aspect-oriented extension to Predicate/Transition (PrT) nets, as a rigorous formalism for threat-driven modeling and verification.</td>
</tr>
<tr>
<td>[8]</td>
<td>Aspect-Oriented Risk-Driven Development (AORDD)</td>
<td>The AORDD Methodology has two steps that must occur prior to any analysis. Create system functional models, using the UML 2.0 and perform a risk assessment of the system.</td>
</tr>
<tr>
<td>[29]</td>
<td>SIREN (SImple REuse of software requiremeNts)</td>
<td>In order to use the security profile properly, the Elicitation phase must start by studying the threats to the assets of the system.</td>
</tr>
<tr>
<td>[11]</td>
<td>Multilevel Secure LAN</td>
<td>A multidimensional threat model that accounts for developmental and operational phases of system evolution and considers both physical and non-physical threats is presented. Physical threats to the system are explicitly addressed in the requirements engineering process.</td>
</tr>
<tr>
<td>[27]</td>
<td>Misuse Case</td>
<td>Identify critical assets in the system. Define security goals for each asset. Identify threats. Identify and analyze risks. Define security requirements for the threats to match risks and protection costs, preferably aided by a taxonomy of security requirements.</td>
</tr>
<tr>
<td>[28]</td>
<td>SQUARE</td>
<td>SQUARE consists of nine steps: agree on Definitions, identify security goals, develop artifacts to support security requirements definitions, perform risk assessment, select requirements elicitation technique, elicit the security requirements, categorize the security requirements, prioritize the security requirements, and inspect the security requirements.</td>
</tr>
<tr>
<td>Reference</td>
<td>Initiative name</td>
<td>Initiative resume</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>[20]</td>
<td>ModelSec</td>
<td>ModelSec proposes a generative architecture based on a chain of model transformations involving several security models at different levels of abstraction.</td>
</tr>
<tr>
<td>[23]</td>
<td>Secure Tropos</td>
<td>Supports development of IS (information systems) through four phases: early and late requirements, architectural and detailed design. Language application includes three major stages.</td>
</tr>
<tr>
<td>[16]</td>
<td>SPEM, Secure Tropos and PriS</td>
<td>Three main activities of the framework: the Security and Privacy Cataloguing, the Security and Privacy Analysis, and the Selection of Cloud Service Provider. The proposed framework consists of a language and a process that is focused on the requirements engineering stage.</td>
</tr>
<tr>
<td>[19]</td>
<td>Attack trees and Misuse Cases</td>
<td>A pair of controlled experiments was performed. In each experiment, the participants used the two techniques individually on two different threat identification tasks.</td>
</tr>
<tr>
<td>[30]</td>
<td>SEDAWA: Secure Engineering process for Data Warehouses</td>
<td>The secure engineering process is proposed with the purpose of defining secure requirements and transforming them in order to develop secure Data Warehouses.</td>
</tr>
<tr>
<td>[13]</td>
<td>Attack trees and Misuse case</td>
<td>The participants were randomly assigned to one of four experiment groups.</td>
</tr>
<tr>
<td>[24]</td>
<td>Methontology</td>
<td>The life cycle activities of Methontology are composed of specification, conceptualization, formalization, implementation and maintenance activities.</td>
</tr>
</tbody>
</table>

Table 6: Brief description of the papers

From these we found that 11 different methodologies exist within these 15 papers selected, which are:
- KAOS: goal-oriented methodology [4]
- Secure Tropos: Agent-Driven Methodologies [12, 16]
- Aspect-Oriented Risk-Driven Development (AORDD) methodology [8]
- SQUARE: security quality requirements engineering [28]
- Information System Security Risk Management [23]
- Aspect-Oriented Petri Nets [33]
- Attack trees [19, 13]
- Misuse cases [19, 13, 27]
- Practical method based on reusable requirement repository [29]
- Model-Driven Architecture/Security [20, 30]
- Ontology-engineering methodology [24]

4.2 Analysis

Regarding the data obtained in the extraction (Table 6), we found out that the case study type was prevalent among other study types, followed by examples.
Only 2 papers had experimental data and analysis, both of them had the same author [19, 24] (Figure 1).

We were interested in knowing how they applied Security in the proposal, if it was thought as a part of the development process or maybe they just considered it as a software component. We can see in Figure 2 that none of the work we selected added security as if it was a software component. Another fact we discovered is that some initiatives found in the selected papers, 7 of 15 seem to be used within an industrial environment (Figure 3).

Most interesting findings can be summarized in the following lines:

- From 11 different papers [4, 8, 19, 30, 13, 27, 33, 28, 23, 12, 16], we found that different types of modelling languages were used. The most used were: UML, Attack Trees, Petri Nets and I*Tropos based approaches.
- A petri based modelling approach of threats [33] shows the progressive and transversal intrinsic nature of security threats.
- SQUARE [28] delivers a set of artifacts, threats and vulnerabilities, from which is possible to get an impact analysis, the probability of occurring and risk level from it. With this is possible to categorize and prioritize security requirements, which allows to analyze the security requirements of the software from the beginning.
- [29] defines a repository of reusable generic requirements which minimize the time of developing in subsequent projects when using the practical methodology the propose. SIREN extends the MAGERIT methodology, conforms to the ISO 15408 or Common Criteria Framework, and also is based on IEEE standards.
- One approach work on how to prove that security was implemented in a project to the stakeholder [4]. They take into account the “usability” of the software as a part of the integration of stakeholders, to make explicit the security to them.
4.3 Answers to Research Questions

Part of the work is to answer our Research Question in Section 3.1, the answer to them are:

1. What are the existing initiatives or methods of identifying security threats and attacks?
   From these methodologies, papers which identify Security Threats were [27, 19, 13, 20, 16, 12, 23, 28, 11, 4, 8, 33, 29], with a total of 10 different types of techniques. Additionally, these techniques considered the Domain of the problem as a first step into getting security threats. According to the results obtained with this work we can say there are a plethora of methods for identifying security threats. In spite of the results, there is no clear distinction for defining threats and attacks, every article read has at least a different conception of the Threat and Attack concepts. From the initiatives found, we can see that most part of them take into account security related threat within the requirement discipline. Some of them have the business concern as the primary focus, and others are worried about satisfying the functional requirements in the most secure way. In the same way, that there are not that many initiatives that are taken into account in the industrial environment. Finally, we can relate to the work of Whyte and Harrison [32], which states the lack of implementation initiatives within the industry, which is clearly showed in the results above.

2. What threat identification methods are related to Security Tactics?
   With the search and extraction done, we could not find anything related to Security Tactics.

3. What other Mitigation Methodologies exist when developing Software, to conceive Secure Systems?
   As some of the methodologies gave proposals to identify security threats/attacks, not everyone gave mitigation methodologies. In [8] they use AOM (Aspect-Oriented Modeling) methodology and techniques to specify generic security mitigation mechanism models. With a set of these security mechanisms, they perform a trade-off analysis between them to improve the security in the design. The view in the Risk analysis model in [4], compresses goal trees arising from risk responses, thereby making it easier to trace risks to mitigating responses, the requirements they treat, and the countermeasures they refine. Mitigation on web based attacks [24] is based in the complete understanding of the domain and context of information contents to be processed, and the ability to filter them on the basis of their effects on the target web application, with the help of policies; which is done through the process of reasoning and intelligent decision making. [33] uses threat mitigations (security features or assurance techniques) for mitigating specific security threats, PrT net-based aspects is the technique used to model threat mitigations. [27, 19, 13] mentions that a Use Case can be a countermeasure for a Misuse Case, in which the first reduces the misuse cases chance of succeeding; they are called security use cases. [23] uses Risk treatment-related concepts or decisions, requirements and controls defined and implemented to mitigate the risks. A risk treatment is the treatment decision for a risk.
5 Threats to Validity/Limitations of this Study

Throughout the execution of the protocol, we took care to reduce the threats that could invalidate the results obtained. We present those threats identified, and try to explain what we did to mitigate them (or chose to accept).

5.1 Threat of missing Literature

In the beginning we were using web search engines to get the conference and journal papers, like IEEEExplore, ACM, Spring, and others. But we decided not to used them because every one of them had different forms to build a search string; which could lead to different kinds of search. Also we found that some web search platform changed the maximum quantity of keywords, so this gave us a big limitation. We changed our focus to ensure a more stable population, allowing the replication of the study. Also, we focused in journals in order to cover studies reporting long term research in initiatives and accumulated scientific evidence. We decided to do a “manual” search within the Journals we knew and had the access to get the material needed. To obtain all the articles within a Journal, we built a script based on Python to download all the information needed to identify an article within a Journal (URL, Volume, Title, Abstract, etc.). To obtain the selected articles, we used Excel macros to filter according to the abstract, with the keyword obtained by the Population and Intervention. To mitigate the possible threat within the procedure to extract papers from the Journals and the filtering within the excel list made with abstract, we did:

- Executed the script at least everyday, for 2 weeks, to compare and see if the results obtained had not changed.
- Executed the macro, with our keywords every time we got a new list from the execution of the script.
- We observed that there were no changes.

We may have filtered out some studies when selecting papers in a time interval. We mitigate this by looking the earliest date in which software development processes (and in consequence all sort of initiatives too) started to become popular. UML and RUP were created on the nineties, but started to become popular in the early 2000.

5.2 Threat of Selection Bias

The selection of papers during the systematic mapping procedure was done by three researchers; two of them performed the selection process by following our guideline in section 3.6. Both researchers applied the inclusion and exclusion criteria, compared results and resolve conflict following the definitions stated in the research protocol. Both researchers applied the inclusion and exclusion criteria, compared results and resolved conflicts following the definitions stated in the research protocol. A third person was in charge of managing the selection and intervening whenever a resolutions was difficult to make by the other two.

\[^5\] Python scripts, datasheets and macros on: https://github.com/pasilvagh/SM-fondecyt2015
5.3 Threat of Inaccuracy of data extraction

This threat was mitigated by the iterative nature of the execution of the protocol, the peer review applied with the 2 extractors and a third reviewer in case of conflicts.

6 Conclusions

A Systematic Mapping was planned and operated in order to identify technologies for security threats identification and mitigation. We covered 11 main software development and security journals, published between years 2000 and 2015 (July). From a total of 10,838 articles, 127 were analyzed following the research protocol. A total of 15 studies presented technologies of security threats identification and/or mitigation, and evidence of its applicability. A total of 10 different techniques covering threats identification and 8 covering the mitigation of threats were found. All the initiatives were integrated to at least one activity of the Software Development Lifecycle (SDLC), while 7 show signs of being adopted in the industry. The mapping found only 15 studies that covered 11 different initiatives. Only two techniques presented scientific evidence of its results through controlled experiments, while others selected studies presented informal case studies or examples. Some of the initiatives are focused on the Requirement Engineering discipline, using reuse or other techniques to create security requirement that could mitigate the threats related to them. A plethora of methodologies exists that try to take into account security within the development process, but they seem to be just proposals without an opportunity for their evaluation within an industrial setting.

This study allowed us to find a range of articles related to our Research Questions, mainly to elicitation of Security Threat and Mitigations. From this studies allowed us to identify only 2 studies with experimental research approach [19, 13], being both a replications of the same experiment. Both studies differed on the experimental subjects: the first time was done with students and the second, with practitioners, in an industrial setting. We did not found more papers with experimental approaches to validate or explore software security technologies.

Within the methodologies and tools found, they all define security in at least one phase and at least 8 techniques in two phases of software development lifecycle. Although there are many forms in which security could be introduced in the software development, the most repeated form was in the search of requirements or security requirements. Elicitation of threats does not always involve a whole methodology, for example Misuse Case and Attack Trees are techniques contextualized in the requirement analysis activities, but not involved in a whole development methodology. [19, 13] concludes with the experimentation that Attacks Trees have a better usability and identifies more security threats than Misuses Cases, although the threats identified by both were different and complementary. This finding guides us to think that there is a research gap on the comparation of threat identification techniques, in order to assess their impact and complementarity. Most of the studies supported the requirements stage of the SDLC, while eight of them seemed to support other development activities. However, proposals were not clear in how they integrate to the rest of the SDLC. Proposal [30], elicit security requirements based on business and organizational goals, in contrast with the majority who elicits from functional requirements. This guides us to think that the scenario based requirements analysis is more suitable for security analysis, but further research is required to validate this idea.
Finally, studies have shown that there is no common agreement with some word definition or language in the Security Community [12, 16, 27, 13, 24, 23, 8]. Some studies presented security domain models to define the domain in which a threat could be related to others entities, others just used a definition already used by others, either way, we realized that the definition of the term “threat” depends on the methodology and it is not a global concept, as some with poor experience in security could have thought. For example, some would define the term “threat” as a way to misuse something in a way it is not supposed to, others may think that a threat is a temporal sequence of attack steps, or even some could not consider this construct and they only use the term “attack”. This suggest the necessity of a common conceptual model focused on identification and mitigation of security threats, to conform a unified vocabulary for future research in technologies for building secure software systems

7 References

References