A qualitative evaluation of model-based security activities for software development

Erkuden Rios[1], Per Håkon Meland[2], Shanai Ardi[3],
Alessandra Bagnato[4], Jostein Jensen[2] , Wissam Mallouli[5],
Fabio Raiteri[4], Txus Sanchez[1], Inger Anne Tøndel[2], Bachar Wehbi[5]

[1] European Software Institute, Parque Tecnológico 204, 48170 Zamudio. Spain
{erkuden.rios, jesus.sanchez}@esi.es,
[2] SINTEF ICT, Software Engineering, Safety and Security, NO-7465 Trondheim, Norway
{per.h.meland, jostein.jensen, inger.a.tondel}@sintef.no,
[3] Department of computer and information science
Linköpings universitet, SE-58183, Linköping, Sweden
{shaar}@ida.liu.se,
[4] TXT e-solutions S.p.A, Via Al Ponte Reale 5, 16100 Genoa, Italy
{alessandra.bagnato, fabio.raiteri}@txt.it,
[5] Montimage, 39 rue Bobillot, 75013 Paris, France
{wissam.mallouli, bachar.wehbi}@montimage.fr

Abstract. Most of the reoccurring types of security problems can be solved by known mitigations in most software products, preferably as early as possible during development. Representing mitigation knowledge in form of reusable security models will help developers in improving software security and learning from past mistakes. This paper explains six model-based security activities that can be integrated with most existing development processes, along with the methods and results of a qualitative evaluation involving software developers from the industry. The evaluation includes semi-structured interviews and questionnaires based on the Technology Acceptance Model (TAM).

Keywords: model-based security, software development, security engineering, qualitative evaluation.

1 Introduction

As stated by Noopur Davis [1]: "*...over 90 % of software security vulnerabilities are caused by known software defect types. [...] the top ten causes account for about 75 % of all vulnerabilities.*" Therefore, in most systems it is possible to substantially improve security by focusing on common security problems that can be solved in similar ways in most software products. Our approach is to transform up-to-date information on security problems and ways to mitigate them in the form of reusable security models. These models are intended to help developers improve software security and learn from past mistakes, and they can be accessed from within development tools.

The purpose of this paper is to present a set of model-based security activities with their supporting modeling formalisms. These activities were subjected to a qualitative evaluation which we also explain and show the results from. All the work has been performed in the context of the EU project SHIELDS [2], which is about reducing known security vulnerabilities during software development by sharing security models through a centralized repository [3].

The qualitative evaluation explained herein was performed at an early stage of the project in order to get feedback from end-users to improve the security activities, the associated models and the descriptions of both. The evaluation was performed by selecting a set of software developers and security experts from some of the industrial partners of SHIELDS. These were exposed to descriptions and examples of the security activities, along with scenarios that describe the context of the activities execution. The evaluation method is based on semi-structured interviews and a set of questionnaires according to Technology Acceptance Model (TAM) [4].

In the next sections we explain the SHIELDS activities and models that have been target of evaluation. We then describe the qualitative evaluation method and summarize the feedback received from industry. This is followed by a discussion on the methodology and the results. Finally, the paper is concluded along with plans on our future work.

2 The SHIELDS activities

The SHIELDS approach is not intended to be a development process of its own, but proposes six activities that can be integrated with most existing development processes with as little extra overhead as possible.

These activities are shown as rounded rectangles in Fig. 1, where they have been related to what is considered to be the most generic phases in any development process, namely *requirements*, *design*, *implementation* and *testing*. Although the SHIELDS activities are complementary, they are not strictly dependent on each other and can be performed separately. This is similar to the seven touch-points or principles for software security proposed by McGraw [5], but our activities are always based on using security knowledge in the form of security models. We believe that this benefits access and comprehension of the information, as well as the process of sharing security information [6].

We will now briefly introduce these activities, but for a more thorough walkthrough with examples the interested reader should refer to the publicly available report on the SHIELDS Web site *D1.2 Initial SHIELDS approach guide* [7] (which was the theoretical source used during the evaluation). The modeling formalisms mentioned related to these activities are further explained in section 3.

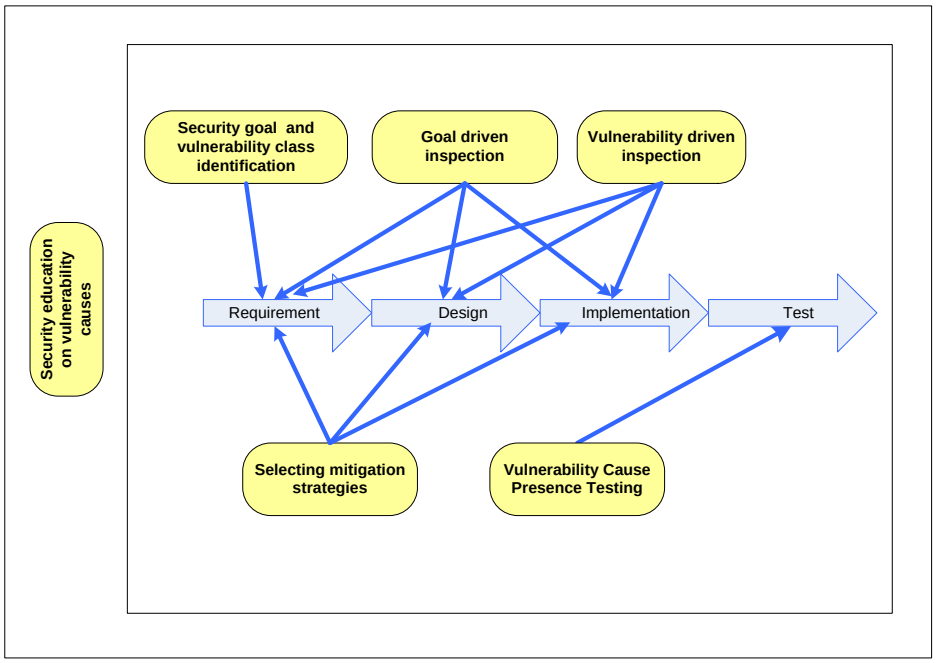

Fig. 1. SHIELDS security activities related to development phases

Security education on vulnerability causes is an activity not directly associated to any specific development phase. Developers are hardly able to design, develop and test secure systems until they understand the most common reoccurring security issues and their causes [8]. An important step toward securing software products is to raise security awareness of developers through the provision of up-to-date information about known vulnerabilities. It is important that developers learn how inadequate requirements, flaws in design, and mistakes in code can result in vulnerabilities. This can be accomplished using *Vulnerability Cause Graph* (VCG) [9], [10], which are graph structures that relate causes to vulnerabilities in a software product. Using VCGs developers can develop an in-depth understanding of vulnerabilities and their causes. This additional depth is crucial to be able to identify solutions to prevent vulnerabilities in future software products and not to repeat same old mistakes.

Security Goal and Vulnerability class identification is an activity that uses threat models to identify both the security goals for the software and the potential vulnerabilities that might occur in it. A developer creates his project specific threat models by starting from generic ones and adding details specific of the project he is working on, or by reusing more specific models from similar development projects and adapting them to the current project needs. The threat models visually present aspects that can be of threat to a software system or a component of a system. These should be used to prioritize what is to be protected (and thus indicate security goals) and identify relevant attacks that exploit commonly found vulnerabilities for this type of system. Additionally, the models can show measures on how to mitigate the

threats. The modeling formalisms we use for threat modeling are attack trees [11] and misuse cases [12].

Goal-driven inspections are manual inspections on different development documents (not just source code) that check for indicators or evidences of the correct implementation of security goals. The starting point for the inspection is a set of identified security goals. A security goal, when met, contributes to meeting some other security goal or ensures that one or more security properties desired by some stakeholder hold. Security goals are closely related to security requirements and policies, but goal-driven inspections can also be performed on the security requirements documents themselves. The technique introduces a model named *Security Goal Indicator Tree* (SGIT) [13] that describes in a tree-like structure the indicators to check for a certain security goal and their relationships. A SGIT is transformable into a *Guided security inspection checklist* [14] that consists of a set of questions for the inspector to answer during the inspection, so that inspection procedure is easily understandable and even non-security experts can perform it.

Vulnerability-driven inspections are manual inspections on different development documents that aim at searching for vulnerability causes. These inspections check for evidences in development documents that indicate that a specific vulnerability is present. In order to guide these inspections for each vulnerability class a *Vulnerability Inspection Diagram* (VID) model is used [15]. The VIDs are a high level description of the inspections and are transformable into *Security Inspection Scenarios* that explain in natural language the manual inspection procedure in even a more understandable way. Both models can be understood by non-security experts so they can perform the inspections.

Selecting mitigation strategies identifies alternative development activities that can be performed in order to prevent vulnerabilities. There are usually a number of alternatives that can be selected to address these security issues. The models that describe the different alternatives are named *Security Activity Graphs* (SAG), which show in a tree-like structure the different alternatives and their combinations. SAGs are used by developers to select the activities that best fits their corresponding development organization [16], [17].

Vulnerability cause presence testing is used to detect vulnerabilities in software products. Here, we also utilize the VCGs by formally defining the information of causes by creating *Vulnerability Detection Conditions* (VDCs) [7], which are then used through testing tools to determine whether the vulnerability is present in the final implementation.

3 The SHIELDS models and formalisms

The modeling formalisms supported by SHIELDS are both newly developed and extensions to previously existing ones. Generally, the introduced extensions have made it easier to add tool support and make the formalisms complement each other through common concepts found in the various models. New formalisms have e.g. been created to support model-based inspection at various stages of the development,

something we did not find covered in existing work. Improvements have also made it possible to show how different types of models are related to each other.

In **Table 1** we present an overview of the main modeling formalisms already mentioned as the driving force of the SHIELDS activities.

Table 1. Overview of SHIELDS models and formalisms.

Modeling formalism	**Purpose**	**Relation to other models**
Misuse case	Get an overview of typical threats towards functionality commonly found in software systems, and common mitigating security activities to these threats.	Provide input for finding relevant attack trees, VCGs, SAGs, SGITs and VIDs.
Attack tree	Get an overview of how an attacker can achieve a specific attack goal, in order to protect a system from such attacks. Attack trees can be used to detail threats in a misuse case.	Provide input for finding relevant VCGs, SGITs and VIDs.
Vulnerability Cause Graph (VCG)	Improve understanding of software vulnerabilities by identifying a vulnerability's causes and their relationships.	Provide Causes and paths leading to a vulnerability that will allow defining VDC. Can identify SAGs.
Security Activity Graph (SAG)	Identify software development activities that can prevent vulnerabilities by addressing their causes.	A SAG is typically associated with a cause in a VCG or mitigation of a threat in a misuse case or attack tree.
Vulnerability Detection Condition (VDC)	Describe system or application behavior in order to detect causes of vulnerabilities in the implementation and execution traces. This information is then typically used by testing tools to perform automated vulnerability detection.	Derived or part of a VCG.
Security Goal Indicator Tree (SGIT)	Describe indicators that can be examined to find if a security goal has been correctly implemented. The structured set of indicators can then be used to guide inspections.	Can be identified from misuse cases and attack trees. Can refer to Security Indicator Specialisation Trees.
Security Indicator Specialisation Tree	Give more details on an indicator that can be used for inspections, e.g. how to check for this indicator in different document types and on different platforms.	Connected to SGITs.
Guided Security Inspection Checklist	Provide an easy to use guide for how to inspect whether a security goal has been correctly implemented.	Used based on SGITs and Security Indicator Specialisation Trees.
Vulnerability Inspection Diagram (VID)	Guide inspections for a specific class of vulnerabilities.	Vulnerabilities can be identified by misuse cases and attack trees.
Security Inspection	Give concrete guidance as to how to perform the actions described in a VID in order to	Identified by VIDs.

Scenario	inspect for the vulnerability.	

4 Evaluation method

In order to get early indications on the perceived usefulness and ease of use of the security activities and related models, we have used a scenarios-based evaluation method. The following text summarizes this method and the results from the evaluation, but for a more thorough explanation, together with the actual questionnaire forms and received answers the readers should refer to the publicly available report *D5.1. Results of First Evaluation of the Technical Work Packages* [18].

The goal of the evaluation was to get professional opinions on the following pre-defined criteria:

- **Level of usability of the activities**, i.e. learning curve, ease of use, efforts in the definition of security requirements and adaptation of existing procedures, expected model interdependencies and lack of coherence.
- **Expected impact** on security and trust of the software produced when performing the activities.
- **Return on Investment (ROI)** that can be expected when adopting the activities in software development processes: expected gain in efficiency, productivity and costs.
- **Potential scalability problems** when used in more complex systems compared to the provided examples.
- **Possibilities of reusing work** (models) from other projects or other users.
- **Possibilities related to adaptations and extensions** to SHIELDS.
- **Compliance with existing development processes** currently used within the organization.

The subjects of the evaluation were provided with two documents that described the descriptions and examples of the activities and models [7] and a set of scenarios showing the larger context of their use [19]. More documents from the SHIELDS project were also made available in case the subjects felt that they needed more details on the technical background of what they were evaluating, but these were not mandatory reading. Additionally, a briefing was made for the participants to explain the expectations from the evaluation and also to present some scenario walkthroughs where example models were showed. During the evaluation, human guidance was also available to assist and clarify any unclear parts of the documents.

The characteristic of the people we wanted for this evaluation were the following:

- Knowledgeable persons from industrial end-users in SHIELDS consortium not having participated in creating what was to be evaluated.
- Security experts familiar with security best practices and tools, as well as practical security reviews, threat analysis and preferably security modeling.
- Experienced software developers that are somewhat knowledgeable of current “best practices” related to secure development.

- People with research experience within the field of software security, with good knowledge of security-related sources of information (such as NVD stats, CERT stats) and various threat level measurers.

Based on this we selected four participants, two from each participating organization. The first organization (*A*) was an SME, while the second organization (*B*) was a larger enterprise. The two organizations are located in different European countries working on somewhat different types of development projects, but both concerned about software security. From organization *A* two participants fit within the description of "Software security expert and researchers (with good knowledge of various security vulnerabilities and an interest in security techniques and models)". From organization *B* the two remaining were characterized as "Software developers (involved in all phases of software development, from specification to maintenance)".

The feedback from the participants was collected using a questionnaire based on the *Technology Acceptance Model* (TAM) [4], followed by an interview performed during a one day session meeting. The interview guide consisted of detailed questions on particular scenarios, questions related to business indicators and improving the SHIELDS activities and models. The questionnaire was tried out beforehand during a pre-test on an independent security expert in order to make sure that they did not lead to any misunderstandings. The interviews were also practiced on beforehand, and performed by two different interviewers. Also four people were involved in the evaluation to analyze the results and draw conclusions.

5 The feedback from the industry

In general, the evaluators (also called subjects of evaluation) found that the security activities and models developed so far have a great potential of usability. To quote one of the subjects; *"SHIELDS technology can be applied in different fields where high system reliability is needed (e.g. industry, telecommunication or medical fields)"*. It was also expressed that the techniques will be useful in *"all development phases, from system design to system implementation, test and monitoring"*.

Nevertheless, subjects also felt that not enough material was given to evaluate the methods completely, and although they agreed that activities will help in the detection and avoidance of vulnerabilities, they did not see clearly if it will be easier to eliminate vulnerabilities. An important point is that they hesitated to state that they would actually use the activities unless it was already an integrated part of their daily development process. In subjects' opinion, the activities should probably be performed by a security expert in the design phase, by a software developer in the implementation phase and by both a security expert and software developer in the testing phase.

To adequately support different types of users it should be possible to easily locate the relevant information (models) that could be used for their particular tasks. Eventually, it should be possible to limit the models and information that is available for a given type of user. The profusion of different model types could make things more difficult to understand for the users. One evaluator suggested that *"it should be possible to say that the more popular development processes are covered by SHIELDS"*.

Related to scalability, it was stated that based on the current documentation it was difficult to see how the activities would perform in complex situations. It was recommended that the SHIELDS activities should be designed so that complex and critical systems can be targeted and more complex examples are presented when describing the activities in the documentation.

As for possibilities for reusing work (models) from other projects or other users, subjects believed that it would be possible to use a centralized database of known security problems and models in future projects. The results of each project can be considered as part of the information/models that the user has to collect to build a complete security database. Additionally, SHIELDS can take profit from the existing major security projects and vulnerability databases. Vice versa, a public API could be created to let other systems use the centralized repository as input. Evaluators also pointed out that statistics should be used to help users find the models and techniques in the repository that are most popular to solve problems that are similar to their own. The effort needed to feed repository with usage statistics should be reduced as much as possible for users or they will be reluctant to provide them.

Concerning the perceived ROI of adopting the activities, evaluators asserted that the prevention of expensive security flaws like loss of data and leak of sensitive information was an important aspect that would make the adoption of the SHIELDS activities profitable and would allow selling better value to customers. About the costs of development, software developers will gain in efficiency and productivity since they can easily find the relevant security information they need in a short period of time. Using the activities should make it faster and simpler to do validation and testing. Product maintenance should also become cheaper, thus improving customer satisfaction.

One of the crucial factors that were pointed out is the necessity to complement the activities with automation tools (at least during implementation and testing). The evaluators believed that special attention should be given to building easy-to-use interfaces for the users and well-defined API's for integrating new tools. One evaluator recommended supporting automatic and periodically analysis of software products to detect whether any new vulnerability is introduced (when modifying the product) or to take into account the new vulnerabilities information added to the centralised repository.

Other suggestions were in the line of improving the description and/or content of some of the usage scenarios or the documentation itself, mostly with examples that show the reusability of the models in other projects and how external vulnerability information databases and security tools can be integrated in the approach.

6 Discussion

There are a number of advantages of having performed a scenario-based evaluation early in this research project, e.g. it helped to improve usability and to eliminate misconceptions and lack of completeness in the activities and modeling formalisms we are working with. Another important issue has been to verify that the scenarios describing the use and context of the activities and models are realistic and achievable

in a real-world setting. The results have given us many indications on how to proceed, such as the need for more complex examples when presenting the models and activities. Simple school-book examples are good for basic understanding, but in our case we need more realistic and detailed ones to show the benefits of adopting the SHIELDS activities in development of complex systems.

The two major factors that reduce the significance of this qualitative evaluation are:

1. The evaluation was mainly based on documentation of the SHIELDS activities and models, but no real trial on using the models or performing the activities during the development of a real application was carried out.

2. Only four people from two software companies participated, which is hardly a number that provides substantial evidence.

Regarding the first factor, we chose to do it this way in order to introduce the end users to the activities and models as early as possible. At the time of the evaluation, the supporting tools were still very immature, which would probably have stolen a lot of the attention of the evaluators. For the next evaluation the evaluators will make a more hands-on test of the SHIELDS activities, models and supporting tools. This will give them a better idea of which aspects of their work will be impacted and how, and we expect to obtain richer feedback.

As for the second factor, the number is low, but we believe that we selected representative candidates, and involving more people would probably not have given us much more fruitful feedback. We saw from the results that the evaluators were pretty much in agreement, which supports this assumption. The next evaluation will of course involve more people so that we can support our work with more evidence and measurements.

7 Conclusion and future work

The qualitative evaluation presented herein was performed during the first ten months of the SHIELDS project (with a total duration of 30 months) on descriptive documentation of the model-based security activities, examples and usage. As a general conclusion from the evaluation, it is believed that adopting these activities will bring benefits to the current software development processes used by the software companies, helping developers to implement reliable software and eliminate vulnerabilities in their products. There is a great interest in easy and efficient solutions to guide developers during their tasks (i.e. conception, implementation and testing) to improve the software security. The interest will be significantly improved if the activities are supported by automation tools (especially during implementation and testing), something which is a major goal of the project, but was not ready for this evaluation.

The next evaluations within SHIELDS are planned for the end of phase 2 (June 2009) and end of the final phase 3 (June 2010). These will be both qualitative and quantitative evaluations aimed at assessing the usefulness and easy of use when actually performing the activities, including the practical appliance of models both

manually and through supporting tools. We will also evaluate creation/modifying models through the use of the centralized model repository.

Acknowledgements

The research leading to these results has received funding from the European Community Seventh Framework Programme (FP7/2007-2013) under grant agreement no 215995. We would also like to acknowledge all the members of the SHIELDS Consortium for their valuable help. Especially we would like to thank Professor Nahid Shahmehri and David Byers from Department of Computer and Information Science at Linköping University for their excellent work in the SHIELDS modelling formalisms and security activities.

References

1. Davis, N.: Developing Secure Software, Software Tech News, vol 8, nr 2, 2005.
2. SHIELDS Project Consortium: SHIELDS Project Homepage, http://www.shields-project.eu
3. Meland, P.H., Ardi, S., Jensen, J., Rios, E., Sanchez, T., Shahmehri, N., Tøndel, I.A.: An architectural foundation for security model sharing and reuse, Proceedings of the Third International Workshop on Secure Software Engineering (SecSE), IEEE Computer Society, Fukuoka, Japan, March 2009.
4. Davis, F.D.: Perceived usefulness, perceived ease of use and user acceptance of information technology, MIS Quarterly 13 (1989) 319–340.
5. McGraw, G.: Software Security: Building Security In, Addison-Wesley, 2006.
6. Ardi, S., Byers, D., Meland, P.H., Tøndel, I.A., Shahmehri, N.: How can the developer benefit from security modeling?, Proceedings of the Second International Conference on Availability, Reliability and Security, ARES2007, IEEE Computer Society, pp. 1017-1025, Vienna, Austria, April 2007.
7. SHIELDS Project: D1.2 Initial SHIELDS approach guide. Report 2009. http://www.shields-project.eu
8. Howard, M.: Building more secure software with improved development process. IEEE Security & Privacy, 2(6):63–65, 2004.
9. Ardi, S., Byers, D., Shahmehri, N.: Towards a structured unified process for software security, Proceedings of the ICSE 2006 workshop on Software Engineering for Secure Systems (SESS06), Shanghai, China, 2006.
10. Byers, D., Ardi, S., Shahmehri, N., Duma, C.: Modeling software vulnerabilities with vulnerability cause graphs, Proceedings of the International Conference on Software Maintenance (ICSM06), Philadelphia, USA, September 2006.
11. Schneier, B.: Attack Trees, Dr. Dobbs Journal, December 1999.
12. Sindre, G., Firesmith, D., Opdahl, A. L.: A reuse-based approach to determining security requirements. In Proceedings of the 9th international workshop on requirements engineering: foundation for software quality (REFSQ'03), Klagenfurt, Austria, 2003.
13. Peine, H., Jawurek, M., Mandel, S.: Security Goal Indicator Trees: A Model of Software Features that Supports Efficient Security Inspection. HASE 2008: 9-18
14. Elberzhager, F., Klaus, A., Jawurek, M.: Software Inspections Using Guided Checklists to Ensure Security Goals, Workshop on Secure Software Engineering, Fukuoka, Japan, 2009.

15. SHIELDS Project: D4.1. Initial specifications of the security methods and tools. Report 2009. http://www.shields-project.eu
16. Byers, D., Shahmehri, N.: Prioritisation and Selection of Software Security Activities, Fourth International Conference on Availability, Reliability and Security, ARES 2009 (IEEE Computer Society ed.), Fukuoka, Japan, March 2009.
17. Byers, D., Shahmehri, N.: A cause-based approach to preventing software vulnerabilities, Proceedings of the International Conference on Availability, Reliability and Security (ARES08), Barcelona, Spain, March 2008.
18. SHIELDS Project: D5.1. Results of First Evaluation of the Technical Work Packages. Report 2009. http://www.shields-project.eu
19. SHIELDS Project: D1.1. Initial architecture and requirements specification. Report 2009. http://www.shields-project.eu