Experimental Evaluation of Secure Software Methodologies using Patterns

EDUARDO B. FERNANDEZ, Florida Atlantic University
HERNAN ASTUDILLO, Universidad Técnica Federico Santa María

Security patterns, in spite of being useful artifacts to build secure systems, are not used in industry as much as they should. One of the reasons is that there are almost no empirical evaluations of their value for building secure systems. We discuss here what has been done on this topic and what new experiments are necessary to prove the value of security patterns. We consider what we should measure in these experiments and how to organize the evaluations. We believe that it is important to evaluate methodologies, not artifacts in isolation and we consider a possible experiment for this purpose. Software architects have proposed tactics to build secure architectures and their combination with patterns can be particularly effective and worthy of a comprehensive evaluation. Lightweight methodologies based on extended tactics should also be considered for evaluation.

Categories and Subject Descriptors: D.2.11 **[Software Engineering]** Software Architecture – Patterns; D.5.1 D.4.6 **[Security]**

General Terms: Design

Key Words and Phrases: security patterns, tactics, empirical software engineering,

1. INTRODUCTION

Although considered as a good way to build secure systems, security patterns have not lived up to their expectations, they are still rarely used in industry. There are several reasons for it. One of them is that are not considered easy to apply, although this is not true in general, several methodologies based on patterns exist to build secure systems in a systematic way (Uzunov et al. 2012). A more compelling reason is that there are almost no empirical tests of their value for building secure systems.

Patterns, or other artifacts such as aspects or tactics, do not have much value in isolation; they need to be part of some methodology that guides the designer through all the stages of building a new system. There are close to 200 published security patterns and to make things worse, these patterns are described in different ways in different catalogs; without a methodology designers would be lost about which pattern to use in a specific situation. This means that experiments intended to validate the isolated use of a specific set of patterns are not meaningful, we need to validate methodologies that use them to build secure systems. In our discussion we emphasize methodologies that use patterns but of course most of our comments apply to any secure software development methodology.

Building secure systems requires a holistic approach, considering all lifecycle stages and all architectural levels. Approaches which start in the design stage, e.g., those described in most software architecture books and papers, do not lead to the most secure system; a secure architecture while being an important factor to the security of the systems is not enough. Discussions of security in the software architecture literature, e.g. (Harrison and Avgeriou 2010), start assuming that the requirements already exist but we see software development as a continuum not as a set of disjoint stages. This means that it does not make sense to test isolated stages or artifacts. Aspects such as conceptual modeling and design cannot be separated because design is a refinement of the conceptual model, not an independent stage. The exception is the requirements stage (threat enumeration) that can be validated on its own because it is the first development stage.

We analyze here what experiments we could perform to show the value of security patterns and how those experiments should be organized, i.e., what features we need to measure and how to measure them. Section 2

Authors' addresses: E.B.Fernandez, Department of Computer and Electrical Engineering and Computer Science. Florida Atlantic University 777 Glades Rd. Boca Raton, FL 33431; email: fernande@fau.edu. Hernan Astudillo, Dept. of Informatics, Universidad Técnica Federico Santa María, Valparaíso, Chile; email: hernan@inf.utfsm.cl

considers what specific features should be measured while Section 3 discusses how to organize the experiments. Section 4 presents our recommendations, and we look at some past evaluations in Section 4. We end with some conclusions.

2. WHAT TO MEASURE?

An important issue in experimental work is what specific quality factors should be measured. When evaluating a methodology to build secure systems we can measure usability, speed to make decisions, security of the final system, efficiency in the use of artifacts or in making decisions, and flexibility to accommodate a variety of policies and changes. Even if a methodology produces systems which have a high level of security it would not be accepted in practice if it is hard to use by typical practitioners. We should ask who are these practitioners? In many places software systems are built by a team of developers guided by a software architect. In smaller places no architect is used and a group of developers must build the complete system. The software expertise of these roles is another variable to consider. In general, an important point behind patterns is that they reflect the experience and expertise of others so they could be used by designers with reduced security knowledge; this is not true for other artifacts such as tactics.

Clearly, the most important feature is the security of the resulting design, a very easy and fast methodology which resulted in insecure designs would not be of much value. The problem is that there are no simple and accepted ways to measure security. Existing experiments focus on the time to produce a solution or completeness of a solution but not on the degree of security for lack of an appropriate metrics; we think that our approach to measuring the degree of security (Fernandez et al. 2010) can be used for this evaluation although it needs more elaboration. It is possible of course, to evaluate methodologies by analyzing each stage and doing a logical evaluation, as done in (Uzunov et al. 2016). For experimental evaluations however, we can use some examples prepared by experts and see how close to their solution are the solutions produced by the participants in the experiment.

Different types of applications require different degrees of security. Instead of a methodology that produces the most secure system but requires high expertise and time we can settle for agile methodologies that produce lower degrees of security but can be used by less experienced developers and/or take less time. As far as we know, none of the methodologies we surveyed has a light/agile version.

Although we want to measure methodologies and not artifacts, there are variations in methodologies which are important for their use and practical acceptance. Some methodologies are general and apply to any system, others are intended for distributed systems, others for cloud-based systems, and so on. It might be easier in some cases to evaluate a specialized methodology.

Although we would want to measure the typical software development stages we can go even farther, considering domain model validation. We can try to show that the patterns in some domain model describe a coherent and complete set of functions. We can also try to show that using reference architectures (RAs) can accelerate building a secure system. Combinations of artifacts, e.g. tactics and security patterns also make sense.

3. ORGANIZING THE EXPERIMENTS

The participants in an experiment must be provided with some training on the use of the methodology. If they don't understand it well it is unlikely they will use it properly and will bias the results. We do not mean they need to be experts on security but they need to understand the intention in the application of patterns.

For a methodology based on patterns a good catalog of security patterns is critical. How this catalog is organized and how detailed are its corresponding pattern descriptions are important issues for the results of the experiment. Some authors extend the existing descriptions; for example, (Yskout et al. 2012) uses annotations in addition to the ones in a standard catalog, but in general it is not clear what to add. On the other hand, some authors consider only the solution sections of a pattern, and neglect the rest of the pattern description; however, without the complete description it is easy to choose a wrong pattern. In fact, the whole idea of a pattern is that it is not just a solution model but it has a set of sections with necessary information.

The complexity of the system to be built for the experiment is another important parameter. A simple system will not show clear differences between systematic methodologies and ad hoc approaches. The value of patterns comes from their abstraction properties that make them suitable to handle complex systems. Experiments can be instrumented to collect statistics that help evaluate factors of interest.

Another variable is the use of students versus the use of industry practitioners. It is clear that the methodologies are intended to finally be used by practitioners but it is very difficult for university researchers to do experiments with them, for obvious reasons. We know of only one experiment conducted with professional developers (Karpati et al. 2014). Since many students already have some practical experience, this may not be a serious disadvantage.

Finally, some experiments compare two methodologies against each other. It seems better to compare participants using some specific methodology against participants using ad hoc approaches. This latter group becomes then a control group and we can show then the advantages of using a systematic methodology.

4. SELECTING AN EXPERIMENTAL EVALUATION

Systematic approaches, particularly those implying some sort of process aligned with the software development life-cycle, are called security methodologies. There are a number of security methodologies in the literature, of which the most flexible and most satisfactory from an industry adoption viewpoint are methodologies that encapsulate their security solutions as security patterns. We have proposed one of these, ASE, described in (Uzunov et al. 2015). ASE considers first the use cases of a system during Requirements analysis, and how these use cases can be subverted (Security Requirements Determination (SecReq) phase, Adversary Modeling (AdvMod) stage). This leads to an identification of countermeasures appropriate at this early stage of development and their modeling into the system's conceptual model. A similar sequence of activities is performed during Design, this time with respect to the system's architecture, considering in detail areas of functionality, threats, appropriate countermeasures, and then modeling the countermeasures to produce a secure software architecture. The remaining activities are concerned with verification (at different stages) and security implementation, which usually implies selecting COTS components. ASE is an improved methodology based on our early work (Fernandez 2013).

Using as reference our secure systems development methodology, we can see that it is possible to do experiments to evaluate each stage and each task. The stages can be tailored individually, and the result of stage-oriented experiments can be used to do this tailoring. We can also evaluate the effect of using a general methodology to include any type of distribution, e.g., web services or clouds, versus designing a methodology specific to the use of some approach to distribution.

An important aspect not mentioned in most papers is that it is not enough for a methodology to select an abstract pattern, e.g. authentication, but the designer needs to choose what type of authentication to use, such as biometric, certificate, or password, and where to place the corresponding service. Patterns have several architectural levels, often only the application level is considered, which does not produce the most secure system. The description included in a pattern template indicates the criteria to select different types of a security mechanism, e.g., should we use symmetric or asymmetric encryption? What class operations should be logged?

Architectural tactics are "measures" or "decisions" taken to improve some quality factor, a definition later refined to "architectural building blocks from which architectural patterns are created" (Bass et el. 2012). Each tactic corresponds to a design decision with respect to a quality factor; i.e., tactics codify and record best practices for achieving that factor but do not prescribe any artifact to realize them. Security patterns carry much more information and can be used in all architectural levels, while tactics are general suggestions and only apply to the software architecture level. Tactics and patterns are indeed intended for different roles, patterns are for inexperienced developers or architects without much knowledge of security, tactics are for software architects who know about security and for systems where security is not the determinant quality factor. It would be interesting to compare these two groups with respect to the quality of their designs. A problem in this comparison is that tactics don't say much about the lower architectural levels while patterns do, so a design based only on tactics will always be less complete than a design based on patterns.

In our work on methodology we have not considered tools, which are an important way to make a methodology more acceptable in industry. This means that, in addition to evaluations, we need to build appropriate tools to support the use of security patterns.

5. RELATED EXPERIMENTAL WORK

Setting up an experiment requires a considerable amount of work and only a few experiments about security have been performed, we review some of them below.

(Yskout et al. 2012) described an experiment using 90 students. The system design and its threats were given so the work of the students was to find a way to mitigate them. The objective was to see if extending pattern descriptions was a useful idea. The problem is that the initial pattern descriptions were taken from (Steel et al. 2005), which are not the most appropriate for a general evaluation because they include only Java-oriented patterns. They did not consider either patterns that must be selected together, e.g., authentication, authorization, and logging usually need to go together. According to them, extensive training on patterns is important to use them well. The study did not use any methodology to build secure systems but analyzed the use of isolated patterns.

(Yskout et al. 2015) also considered the use of patterns in isolation. In this experiment the designers used only 36 patterns, which made their work much simpler but reduced the solutions to only those which are commonly used. If the space of solutions is small there will be less difference between using patterns and finding solutions based on experience. The functional design was given and the students had to perform seven specific tasks to harden it, where their only problem was to find the right security solution. In a real case, the designers have to select different patterns at different design stages and have to consider the complete system. Again, if the tasks are simpler, security patterns, which include a good amount of information intended to guide their use, have less value. Really what these students did was to determine the security requirements of the system. The design stage defines precisely what specific mechanism will be used to realize each abstract pattern. Patterns are very useful in this stage but the experiment did not touch this aspect. The measurements in the project were intended to see if the students corrected a set of predefined misuse cases. In a real situation we do not know the threats and we need a way to enumerate them as part of the methodology to build the secure system.

(Astudillo et al. 2014) described a systematic attempt to compare tactics to security patterns via an experimental study. They analyzed the impact in the quality and effort of design decisions of two variables: the technique used by the participants and the experience of the participants. The study used practicing developers and graduate students, each group including novices and experts; subjects were trained in both techniques, and given a relatively simple problem (a tsunami warning system under development); they measured the rate of effectively addressing threats (quality) and elapsed time to answer (effort). They conjectured that security patterns would improve novices' quality and security tactics would improve experts' speed; however, preliminary results indicate that while experts are better than novices at identifying threats, they are no better at mitigating them. Also these results were evaluated in isolated way, not as part of a methodology.

(Opdahl and Sindre 2009) reported on a pair of controlled experiments that compared two methods for early elicitation of security threats, namely attack trees and misuse cases. The 28 and 35 participants in the two experiments solved two threat identification tasks individually by means of the two different techniques. The dependent variables were effectiveness of the techniques measured as the number of threats found, coverage of the techniques measured in terms of the types of threats found and perceptions of the techniques measured through a post-task questionnaire. The only difference was that, in the second experiment, the participants were given a pre-drawn use-case diagram to use as a starting point for solving the tasks. In the first experiment, no pre-drawn use-case diagram was provided. The main finding was that attack trees were more effective for finding threats, in particular when there was no pre-drawn use-case diagram. However, the participants had similar opinions of the two techniques, and perception of a technique was not correlated with performance with that technique. As mentioned above, the threat identification stage appears to be the only stage that can be studied in isolation. We believe that our approach based on use case activities (Fernandez2013) is superior to using misuse cases.

(Karpati et al.) discussed an experiment where industrial practitioners performed the experimental tasks in their workplace. The industrial experiment confirmed a central finding from the student experiments: that attack trees tend to help identifying more threats than misuse cases. It also presents a new result: that misuse cases tend to encourage identification of threats associated with earlier development stages than attack trees. The two techniques should therefore be considered complementary and should be used together in practical requirements work.

(Massacci) reportedon a controlled experiment conducted with 28 master students to compare two classes of risk-based methods, visual methods (CORAS) and textual methods (SREP). The aim of the experiment was to compare the effectiveness and perception of the two methods. The participants divided in groups solved four different tasks by applying the two methods using a randomized block design. The dependent variables were effectiveness of the methods measured as number of threats and security requirements identified, and perception of the methods measured through a post-task questionnaire based on the Technology Acceptance Model. The experiment was complemented with participants' interviews to determine which features of the methods influence their effectiveness. The main findings were that the visual method is more effective for identifying threats than the textual one, while the textual method is slightly more effective for eliciting security requirements. In addition, visual method overall perception and intention to use were higher than for the textual method.

6. CONCLUSIONS

In addition to the difficulty of choosing experiments it is not easy to define what to measure. We think that instead of trying evaluation of specific artifacts, we should evaluate a complete methodology. However, we should select carefully what methodology to evaluate and what aspects to measure.

Past work does not provide much guidance, three of the methods studied are about finding threats, three studies considered the design stage, and no study considered a complete methodology.

A promising direction to facilitate the adoption of security patterns is to combine them with tactics, already known and accepted by software architects. In recent work we have improved the original set of tactics (E.B.Fernandez et al. 2015b) and we intend to evaluate their use in conjunction with security patterns, using our proposed methodology (Uzunov et al. 2015). Another possibility is to define a lightweight methodology using tactics with improved descriptions that can be used for those applications where security is not so critical. We also need to develop ways to measure the degree of security reached by a system built in this way.

ACKNOWLEDGEMENTS

We thank our shepherd Cédric Bouhours, for valuable suggestions that improved this paper.

REFERENCES

H. Astudillo, E. B.Fernández R. Noël, G. Pedraza, "An Exploratory Comparison of Security Patterns and Tactics to Harden Systems", Procs. of the ESELAW 2014 conference, a track of the 17th Ibero-American Conf. on Soft. Eng.(CIbSE 2014)

L. Bass, P. Clements, and R. Kazman, *Software architecture in practice* (3rd Ed), Addison-Wesley 2012.

E.B.Fernandez, N. Yoshioka, H. Washizaki, and M. VanHilst, "Measuring the level of security introduced by security patterns", *Procs. of the 4th workshop on Secure systems methodologies using patterns (SPattern 2010)*, in conjunction with ARES 2010, Krakow, Poland, February 2010

E.B.Fernandez, “Security patterns in practice: Building secure architectures using software patterns”, Wiley Series on Software Design Patterns, 2013.

E.B.Fernandez, Raul Monge, and Keiko Hashizume, “Building a security reference architecture for cloud systems”, Requirements Engineering, 2015 Doi: 10.1007/s00766-014-0218-7

E.B.Fernandez, H. Astudillo, and G. Pedraza-Garcia, “Revisiting architectural tactics for security”, 9th European Conf. on Software Architecture (ECSA 2015), ACM/IEEE, September 5-7, 2015.

N. B. Harrison and P. Avgeriou, “How do architecture patterns and tactics interact? A model and annotation”, *The Journal of Systems and Software*, 83, 2010, 1735-1758.

P. Karpati, Y. Redda, A.L. Opdahl, G. Sindre, “Comparing attack trees and misuse cases in an industrial setting”, Information & Soft. Technology, Vol. 56, No. 3, March 2014, 294-308.

K. Labunets, F. Massacci, F. Paci, L.M.S. Tran, “An experimental comparison of two risk-based security methods”. In Proceedings of the 7th *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 163–172, 2013.

A. L. Opdahl, G. Sindre, “Experimental comparison of attack trees and misuse cases for security threat identification”, Information and Software Technology, 2009/5/31, Vol. 51, No 5, 916-932

C. Steel, R. Nagappan, and R. Lai, Core Security Patterns: Best Strategies for J2EE, Web Services, and Identity Management, Prentice Hall, Upper Saddle River, New Jersey, 2005

A.V. Uzunov, E.B. Fernandez & K. Falkner (2012), "Securing distributed systems using patterns: A survey", Computers & Security, 31(5), 681 - 703. (ISI, IF= 0.868)), doi:10.1016/j.cose.2012.04.005

A. Uzunov, E. B Fernandez, K. Falkner, “ASE: A Comprehensive Pattern-Driven Security Methodology for Distributed Systems”, Journal of Computer Standards & Interfaces , Volume 41, September 2015, 112-137.

A. Uzunov, E.B.Fernandez, and K. Falkner, “Assessing and Improving the Quality of Security Methodologies for Distributed Systems”, submitted for publication. 2016.

K. Yskout, R.Scandariato, W. Joosen, “Does organizing security patterns focus architectural choices?”, ICSE 2012, Zurich, Switzerland, 617-627.

K. Yskout, R. Scandariato, W. Joosen, “Do Security Patterns Really Help Designers?”, International Conference on Software Engineering (ICSE), Florence, Italy, May 2015