SINTEF

SINTEF A26187 - Unrestricted

Report

Assessing the Usefulness of Testing for Validating the Correctness of Security Risk Models Based on an Industrial Case Study

Author(s)
Gencer Erdogan[1], Fredrik Seehusen[1], Ketil Stølen[1], and Jan Aagedal[2]

[1] Department for Networked Systems and Services, SINTEF ICT
PO Box 124 Blindern, N-0314 Oslo, Norway

[2] Accurate Equity, Martin Linges vei 25, N-1364 Fornebu, Norway

SINTEF ICT
Networked Systems and Services
2014-06-23

SINTEF

SINTEF IKT
SINTEF ICT

Address:
Postboks 124 Blindern
NO-0314 Oslo
NORWAY

Telephone:+47 73593000
Telefax:+47 22067350

postmottak.IKT@sintef.no
www.sintef.no
Enterprise /VAT No:
NO 948 007 029 MVA

KEYWORDS:
Test-driven security risk analysis,
Test-based security risk analysis,
Risk analysis,
Testing,
Security

Report

Assessing the Usefulness of Testing for Validating the Correctness of Security Risk Models Based on an Industrial Case Study

VERSION
Final

DATE
2014-06-23

AUTHOR(S)
Gencer Erdogan, Fredrik Seehusen, Ketil Stølen, and Jan Aagedal

CLIENT(S)
Norwegian Research Council

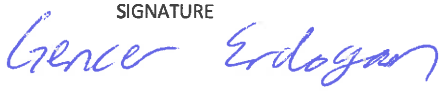
CLIENT'S REF.
201579/S10

PROJECT NO.
102002253

NUMBER OF PAGES/APPENDICES:
16/0

ABSTRACT
We present the results of an evaluation in which the objective was to assess how useful testing is for validating and gaining confidence in the correctness of security risk models. The evaluation is based on a case study where the target system analyzed was a web-based application. The evaluation suggests that the testing was useful in the sense that it yielded new information which resulted in an update of the security risk model after testing.

PREPARED BY
Gencer Erdogan

SIGNATURE


CHECKED BY
Atle Refsdal

SIGNATURE


APPROVED BY
Bjørn Skjellaug

SIGNATURE


REPORT NO.	ISBN	CLASSIFICATION	CLASSIFICATION THIS PAGE
SINTEF A26187	978-82-14-05355-5	Unrestricted	Unrestricted

Table of Contents

1 Introduction

Security risk analysis (SRA) is a process that is carried out in order to identify and assess security specific risks. Traditional risk analysis often rely on expert judgment for the identification of risks and their causes and the estimation of their likelihood and consequence. The outcome of these kinds of risk analysis is therefore dependent on SRA participant's background, experience, and knowledge, which in turn reflects an uncertainty in the correctness of the SRA results.

In order to validate the correctness of the SRA results, the SRA process can be complemented by other ways of gathering information of relevance to the analysis other than relying on expert judgment by the SRA participants. One such approach is to combine risk analysis with security testing following the steps described below:

Phase 1 Establish context and target of evaluation, and carry out security risk analysis of the target of evaluation relying mostly on expert judgment from the SRA participants.

Phase 2 Generate and execute security tests that explore the risks identified during the security risk analysis.

Phase 3 Validate and update the risk model (produced in phase 1) based on the security test results.

We refer to this approach as test-driven security risk analysis (TSR).

Between March 2011 and July 2011 we conducted an industrial case study based on the TSR process. The target system analyzed is a multilingual Web-based e-business application. The system serves as the backbone for the system owner's business goals and is used by a large number of users every day. The system owner, which is also the client that commissioned the case study, required full confidentiality. The results that are presented in this report are therefore limited to the experiences from applying the TSR approach.

From a scientific perspective, our objective with the case study was to assess how useful testing is for gaining confidence in the correctness of the risk models produced in the risk analysis (phase 1 above). To make the analysis precise, we have specifically focused on the degree to which the testing yielded information that caused us to change the risk model.

In the case study, the testing yielded new information which was not found in the risk analysis phase (phase 1 above). In particular, new vulnerabilities were found which resulted in the likelihood values of threat scenarios and risks in the security risk model to be updated. We believe that the update led to a more correct risk model, and that the testing therefore was useful in gaining confidence about the correctness of the risk model.

The rest of the report is structured as follows: Section 2 describes the research method. Section 3 gives an overview of the TSR process used in the case study. Section 4 describes evaluation results. Section 5 provides a discussion of the results. Section 6 discusses related work. Finally, Section 7 concludes the report.

2 Research Method

The basis of our evaluation is to compare the risk models produced before and after testing. Before making this comparison, we first describe what we mean by a risk model, and what we mean by risk models being different.

2.1 Risk Models

Risk models are created during the risk analysis phase. These models contain the identified risks as well as other information that is relevant for the analysis such as the cause of risks and how likely it is that these causes will occur and so on.

The kind of risk models that were produced in our case study were CORAS risk models [10]. These were constructed on-the-fly during a series of workshops. All the information in these workshops were based on expert judgment, mostly from the customer on whose behalf the analysis was conducted.

As illustrated by the example in Fig.1, a CORAS risk model is a directed acyclic graph where every node is of one of the following kinds:

Threat A potential cause of an unwanted incident.
Threat scenario A chain or series of events that is initiated by a threat and that may lead to an unwanted incident.
Unwanted incident An event that harms or reduces the value of an asset.
Asset Something to which a party assigns value and hence for which the party requires protection.

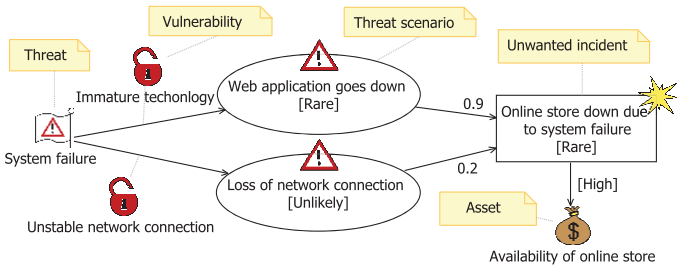


Fig. 1. *Example of a CORAS risk model.*

Risks can also be represented in a CORAS risk model. These correspond to pairs of unwanted incidents and assets. If an unwanted incident harms exactly one asset, as is the case in Fig. 1, then this unwanted incident will represent a single risk. In the case study, each of the identified unwanted incidents harmed exactly one asset. Thus every unwanted incident corresponded to exactly one risk. For this reason, we will only use the CORAS models with unwanted incidents as

basis for the evaluation, and we will sometimes use the terms unwanted incident and risk interchangeably when the distinction is not important.

A risk may have a *risk level* which is calculated from the combination of a likelihood value and a consequence value. For instance, the risk corresponding to the unwanted incident in Fig.1, has a risk level which is calculated from the combination of the likelihood value *Rare* and the consequence *High*.

The risk levels that were used in the case study were calculated by using *risk matrices* (one for each asset) assigning to every pair of likelihood and consequence value (w.r.t. a particular asset), one of the three values: Low, Medium, and High.

Risk The likelihood of an unwanted incident and its consequence for a specific asset.

A relation in a CORAS model may be of one of the following kinds:

Initiates relation going from a threat A to a threat scenario or an unwanted incident B, meaning that A initiates B.

Leads to relation going from a threat scenario or an unwanted incident A to a threat scenario or an unwanted incident B, meaning that A leads to B.

Harms relation going from an unwanted incident A to an asset B, meaning that A harms B.

Relations and nodes may have assignments, in particular:

Likelihood values may be assigned to a threat scenario or an unwanted incident A, estimating the likelihood of A occurring.

Conditional probabilities may be assigned on the leads to relations going from A to B, estimating the probability that B occurs given that A has occurred.

Consequence values may be assigned on the harms relations going from A to B, estimating the consequence that the occurrence of A has on B.

Vulnerabilities may be assigned on the leads to relations going from A to B, describing a weakness, flaw or deficiency that opens for A leading to B.

Throughout this document, we use the term *risk model element*, or just *element* for short, to mean either a node, a relation, or an assignment.

2.2 Difference between Risk Models

Two CORAS risk models are equal if they contain the same risk model elements. Otherwise they are not equal. Let RM_B be the risk model before testing, and RM_A be the risk model after testing, then we distinguish between 3 different kinds of changes

Add An element in RM_A has been added if it is not in RM_B.

Delete An element in RM_B has been deleted if it is not in RM_A.

Edit A node or relation in both RM_B and RM_A has been edited if its assignment in RM_B or RM_A is different.

3 Overview of Process for Test-driven Security Risk Analysis

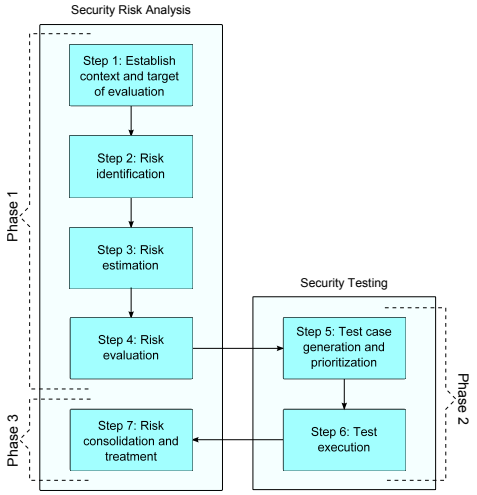


Fig. 2. *The steps in Test-driven Security Risk Analysis.*

This section presents the process according to which the case study was carried out.

As shown in Fig. 2, our TSR process is divided into three phases which in turn are split into various steps. Phase 1, the risk analysis phase, consists of four steps corresponding to steps of the CORAS risk analysis method [10]. These are the steps in which the risk model, which serves a basis for test identification, are constructed. In the case study, the risk models were mainly constructed in workshops by the analyst team with help from the customers. The reader is referred to [10] for a more detailed explanation of the steps of phase 1.

Phase 2 consists of two steps, test case generation and prioritization (step 5), and test execution (step 6). Step 5 was performed as follows. First we made a list of all the threat scenarios identified in the risk analysis phase. Then we selected those threat scenarios that were considered most important to test (i.e., we did not test all the threat scenarios of the risk model). Typically, a threat scenario would correspond to a particular type of attack, and the purpose of testing a threat scenario was to find vulnerabilities that could be exploited by the attack. The priority of a threat scenario was assessed by considering three things:

Severity An estimate of the impact that a threat scenario has on the identified risks of the analysis.

Testability An estimate of the time it would take to test a threat scenario and/or whether the threat scenario could be tested given the tools available.

Uncertainty An estimate of the uncertainty related to the severity estimate of a threat scenario. High uncertainty suggests a need for testing.

After selecting the threat scenarios with the highest priority, we made a detailed description of how we intended to test them.

Step 6, test execution, is the activity of carrying out the tests. In the case study, the tests were executed using tools for automated, semi-automated, and manual testing:

Automated testing IBM Rational Software Architect (IBM RSA) [2], Smartesting CertifyIt [5], Selenium [4], Eclipse [1];
Semi-automated testing OWASP WebScarab[3];
Manual testing Wireshark [6];

The automated process was as follows. First we specified a model of the system under test using IBM RSA together with the CetifyIt plugin, then we used CetifyIt to generate abstract Java tests, then we concretized these in Java, and finally we executed them using Selenium.

All the tests were executed at the level of the HTTP protocol (i.e., at the application level) and from a black-box view of the web-application that was analyzed.

Phase 3, consisting of step 7, has two main purposes. The first is the update the risk model (which is the output from phase 1) based on the test results (which is the output of step 6). The second is to identify treatments for the updated risk model. This step was based on the corresponding step of the CORAS method. Most of the reminder of this paper will be devoted to explaining the degree to which we had to update the risk model based on the test results.

The case study was carried out through 6 workshops. An overview of these workshops is given in Table 1. In the participants column, the denotations C:n and A:m represents n participants from the customer side and m participants from the risk analysis team. Notice that there was no workshop addressing step 6 (test execution) of the TSR process. This is because this step was entirely conducted between the fifth and the sixth workshop.

Table 1. *Overview of the workshops.*

Date	**Step**	**Participants**	**Duration**
28.03.11	Step 1	C:1, A:3	2 hours
12.04.11	Step 1	C:3, A:4	3 hours
09.05.11	Step 1	C:2, A:3	3 hours
20.05.11	Step 2 and 3	C:1, A:2	6 hours
27.05.11	Step 4 and 5	C:1, A:2	6 hours
07.07.11	Step 7	C:2, A:2	2 hours

4 Results

In this section, we describe the differences between the risk models before and after testing.

After testing, no nodes or relations were added to or deleted from the risk model. The only risk model elements that were added or deleted were vulnerabilities (recall that vulnerabilities are not classified as nodes, but as assignments to relations). More precisely, one vulnerability was deleted and four vulnerabilities were added after testing. In Fig. 3, we have illustrated both the number of risk model nodes and the number of vulnerabilities before and after testing.

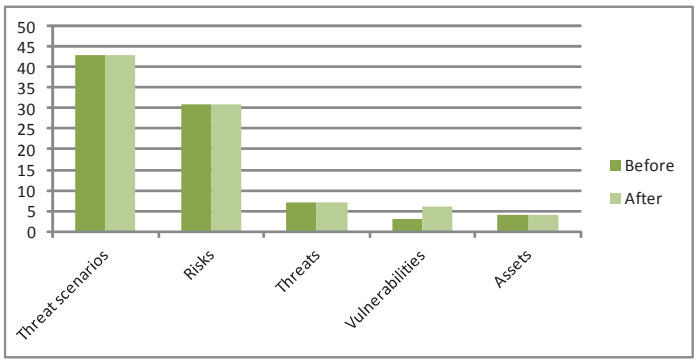


Fig. 3. *Number of risk model nodes and vulnerabilities before and after testing.*

The only risk model elements that were tested in the case study were threat scenarios. The aim of testing a threat scenario was to discover whether the target of evaluation had any vulnerabilities that could be exploited by the attack described by the threat scenario.

In Fig. 4, we show the total number of threat scenarios and risks that were identified, and how many of these were tested. We have also shown the number of *potentially affected* unwanted incidents or threat scenarios that the tested threat scenarios could lead up to. All threat scenarios or unwanted incidents that a tested threat scenario T can lead up to are potentially affected by the testing, e.g., if the likelihood value of T is edited after testing, then this could potentially cause the likelihood values of all risk elements led up to by T to be edited as well. As can be deduced from Fig. 4, 14% of the threat scenarios were tested, 33% of the threat scenarios were potentially affected by the testing, and 42% of the risks were potentially affected by the testing.

Fig. 5 shows the difference between the threat scenarios and risks before and after testing for all risk model elements that were tested or potentially affected by the testing. In the figure, each threat scenario and risk TR has a label of the form i/j which means that TR had a likelihood value of i before testing, and j after testing. The likelihood scale that was used in the case study can be mapped

to a number between 1 and 5, where 1 represents the most unlikely value and 5 denotes the most likely value.

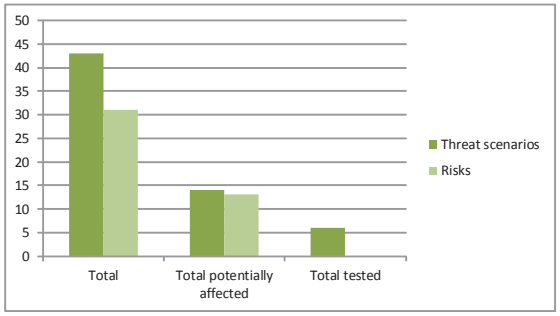


Fig. 4. *Number of risks and threat scenarios tested and potentially affected by the testing.*

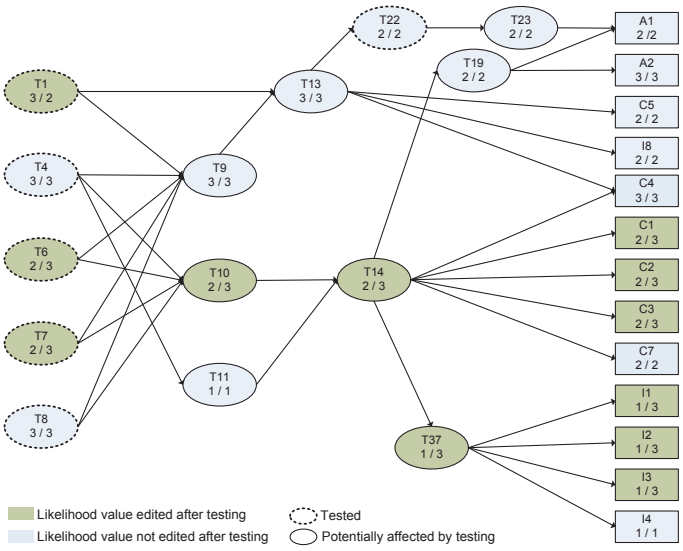


Fig. 5. *Difference between risk models before and after testing.*

All the threat scenarios and risks whose likelihood values were edited after testing are in Fig. 5 represented with a darker color (green) than those threat scenarios and risks that were not edited (blue). Note that all except one risk element whose likelihood values were edited after testing were estimated to be more likely after testing than before testing.

In Fig. 5, the threat scenarios that were directly tested are represented by ellipses with a dotted outline; all the other elements of the diagram are potentially affected by the tested threat scenarios. It can be noted that the level of indirection from the tested threat scenarios to the risks is quite large.

Recall that risks may have risk levels which are calculated based on likelihood and consequence values. The change in likelihood values resulted in the risk levels of four risks being changed after testing. All risks whose risk levels changed were given a higher risk level after testing than before testing. In Fig. 6, we have illustrated the total number of risks identified, the number of risks that were potentially affected by the testing, and the number of risks whose risk levels were edited after testing.

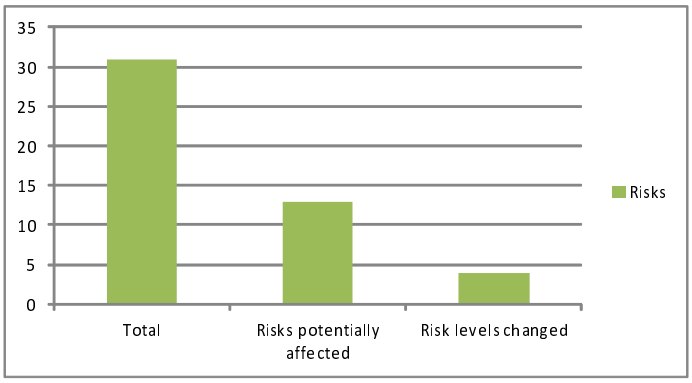


Fig. 6. *Risk levels changed.*

5 Discussion

5.1 The Need to Add Elements after Testing

Based on the discussion in Sect. 4 and the numbers in Fig. 3, we know that new vulnerabilities were added to the risk model after testing, and that no other kinds of risk elements were added.

Why did the testing only yield new information about the vulnerabilities? The main reason for this is that the tests were designed from the threat scenarios, and the purpose of the tests were to identify vulnerabilities that could be exploited by the threat scenarios. In other words, the tests were designed to uncover vulnerabilities; not unknown assets, threats, threat scenarios, or risks. These elements were instead part of the context in which the testing was performed.

Recall that an asset is something that is of value for the party, and that can be harmed by a risk. If a party has no assets, then there is no reason to conduct

a risk analysis. For this reason, assets are always identified in the beginning of the risk analysis, before the risks are identified. In our experience, the process of identifying the risks has never led to the identification of new assets because the assets are then part of the context of the risk identification. The same is also true for the testing.

The argument is similar regarding threats. A threat is a potential cause of an unwanted incident such as a hacker, an insider or a virus, and the testing is performed *with regards to* the identified threats. It therefore seems unlikely that the testing would uncover additional threats.

In principle, we cannot rule out that it would be possible that the test results could yield information that would lead to the identification of new threat scenarios or risks. For instance, it might be the case that a threat scenario may be refined (i.e., split up into more than one threat scenarios) after testing, or lead to the identification of an unwanted incident that had not been previously thought of. However, as long as the tests are designed to uncover vulnerabilities, we believe that this would be unlikely.

It is worth noting that vulnerabilities uncovered by testing in the case study could never have been uncovered if we had performed a risk analysis alone (without doing the testing), regardless of how much effort we would have spent. This is because the testing uncovered issues which only appeared in extremely specific circumstances which could not have been reproduced without executing the system under analysis.

5.2 The Need to Delete Elements after Testing

Based on the discussion in Sect. 4, we know that the testing resulted in the deletion of exactly one risk element – a vulnerability. Furthermore, we believe that this result is generalizable. That is, in general, threats, threat scenarios, risks and assets are unlikely to be deleted after testing, whereas vulnerabilities may be deleted.

The reason why we deleted a vulnerability after testing was that the testing provided evidence that a potential vulnerability identified in the risk analysis phase was actually not present in the system. This led us to remove the vulnerability from the risk model. We also believe that in general, testing can result in the deletion of vulnerabilities, since the tests can be designed to check whether a vulnerability is actually present in the system or not.

The reason why threats and assets are unlikely to be deleted after testing is the same as for why they are unlikely to be added after testing. That is, the assets and threats are part of the context in which the testing is performed, and the testing is therefore unlikely to yield information about this context.

As for threat scenarios and unwanted incidents, these are risk model elements that contain assigned likelihood values. Therefore, there will never be a need to delete these from the risk model after testing. Instead of deleting them from the risk model, we would assign a low likelihood value on these risk model elements.

5.3 The Need to Edit Elements after Testing

As documented by the figures of Sect. 4, 14% of the likelihood values of threat scenarios, 19% of the likelihood values of unwanted incidents, and 13% of the risk levels of risks were edited after testing.

For all risk elements that were edited (with the exception of one), the likelihood value was increased after testing, i.e., the risk element was believed to be more likely after testing than before testing. The reason for this was that the testing uncovered vulnerabilities that were previously unknown, and that led to the belief that certain threat scenarios were more likely to occur than believed before testing.

For one of the threat scenarios, the likelihood value was decreased after testing as a result of one vulnerability being deleted.

In general, we believe that testing will uncover information that may cause the likelihood values of threat scenarios and unwanted incidents to be edited after testing.

The testing did not result in the consequence values that unwanted incidents have on assets being edited. The reason for this is that all the tests were designed to uncover information about vulnerabilities that would increase or decrease the likelihood of a successful attack; the consequence of a successful attack was already known in advance. Is this result generalizable? We believe it is. As long as all the risks have been identified before testing, their consequences can be estimated before testing, and it is unlikely that the testing will uncover information which will cause the consequence values to change.

6 Related Work

This paper has addressed test-driven risk analysis (TSR), i.e., the process of using testing to improve the risk analysis results. We distinguish this from the notion of risk-based testing (RBT) which is the process of using risk analysis to improve the test results. This distinction is usually unclear in the literature. However, most of the literature on combining risk analysis and testing fits into the latter category (RBT). Indeed, the idea of combining risk analysis and testing first originated from the testing community [8]. Since then, a number of approaches to RBT have been proposed. These approaches have in common that the identified risks become the central factor used for a systematic identification of test objectives, the evaluation of the test quality and the test results. Redmill [11, 12], for example, describes the use of risk analysis to optimize the test planning. The quantified risks are used to prioritize the testing and the analysis of system failures and their impact. Amland [7] calculated the risk of the individual software functions and the expected failure probability for these functions weighted with risk factors such as design quality, size and complexity of the evaluated software. Finally, Stallbaum et al. [13] describe a model-based approach, which allows the automatic derivation of system test cases from different kind of functional models as well as their prioritization based on risk. This

approach is called RiteDAP (Risk-based test case Derivation And Prioritization) and uses activity diagrams as system models to automatically generate test case scenarios. Those test models are augmented with the risk values needed for the test case prioritization.

Although some RBT approaches exist, we are not aware of any approach for TSR – let alone any that target security. One of the reasons for this may be that many risk analysis process standards such as ISO 31000 [9] are so generic that they do not stipulate the manner in which data for the risk analysis is collected and used. Therefore, the notion TSR can be seen as an instance of many risk analysis processes.

7 Conclusions

We have described an evaluation of a process for test-driven security risk analysis (TSR) based on our experiences from applying this process in a case study. The objective of the evaluation was to evaluate how useful testing is in gaining confidence in the correctness of the risk models produced in the risk analysis phase of the TSR process. To make the evaluation precise, we analyzed the difference between the risk model produced before testing and the risk model produced after testing.

The process of testing yielded information which led to a change in the risk model created before testing. Specifically, four vulnerabilities were added to the risk model, one vulnerability was deleted, and 14% of the likelihood values of threat scenarios, 19% of the likelihood values of unwanted incidents, and 13% of the risk levels of risks were edited after testing.

We believe that the testing was useful in the sense that it yielded a more accurate risk model. But more than this, the testing uncovered vulnerabilities that would never have been uncovered in the risk analysis phase, regardless of how much effort we would have spent. In other words, if the testing phase had been dropped and we instead had extended the risk analysis phase with the corresponding effort, we would not have uncovered the vulnerabilities that were uncovered in the testing phase.

Acknowledgments. This work has been conducted as a part of the DIAMONDS (201579/S10) project funded by the Research Council of Norway, as well as a part of the NESSoS network of excellence funded by the European Commission within the 7th Framework Programme.

References

1. Eclipse. `http://www.eclipse.org/`. Accessed November 7, 2011.
2. IBM Rational Software Architect (RSA). `http://www.ibm.com/developerworks/rational/products/rsa/`. Accessed November 7, 2011.
3. OWASP WebScarab. `https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project`. Accessed November 7, 2011.

4. Selenium - Web Browser Automation. `http://seleniumhq.org/`. Accessed November 7, 2011.
5. Smartesting CertifyIt. `http://www.smartesting.com/index.php/cms/en/product/certify-it`. Accessed November 7, 2011.
6. Wireshark. `http://www.wireshark.org/`. Accessed November 7, 2011.
7. S. Amland. Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study. *Journal of Systems and Software*, 53(3):287–295, 2000.
8. J. Bach. Heuristic risk-based testing. *Software Testing and Quality Engineering Magazine*, 11:9, 1999.
9. International Organization for Standardization. *ISO 31000:2009(E), Risk management – Principles and guidelines*, 2009.
10. M. S. Lund, B. Solhaug, and K. Stølen. *Model Driven Risk Analysis - The CORAS Approach*. Springer, 2011.
11. F. Redmill. Exploring risk-based testing and its implications: Research articles. *Softw. Test. Verif. Reliab.*, 14(1):3–15, March 2004.
12. F. Redmill. Theory and practice of risk-based testing. *Software Testing, Verification and Reliability*, 15(1):3–20, 2005.
13. H. Stallbaum, A. Metzger, and K. Pohl. An automated technique for risk-based test case generation and prioritization. In *Proc. of the 3rd international workshop on Automation of software test*, pages 67–70. ACM, 2008.

Technology for a better society

www.sintef.no