

Report

An Evaluation of a Test-driven Security Risk Analysis Method Based on an Industrial Case Study

Author(s)

Gencer Erdogan
Fredrik Seehusen
Yan Li

SINTEF IKT
SINTEF ICTAddress:
Postboks 124 Blindern
NO-0314 Oslo
NORWAYTelephone:+47 73593000
Telefax:+47 22067350
postmottak.IKT@sintef.no
www.sintef.no
Enterprise /VAT No:
NO 948 007 029 MVA

Report

An Evaluation of a Test-driven Security Risk Analysis Method Based on an Industrial Case Study

KEYWORDS:Risk analysis
Testing
Security
Security risk analysis
validation**VERSION**

Final

DATE

2013-12-03

AUTHOR(S)Gencer Erdogan
Fredrik Seehusen
Yan Li**CLIENT(S)**

Norwegian Research Council

CLIENT'S REF.

201579/S10

PROJECT NO.

102002253

NUMBER OF PAGES/APPENDICES:

24/0


ABSTRACT

This report is an evaluation describing the experiences obtained from a case study, carried out in a period of eight months from June 2012 to January 2013, in which we conducted a test-driven security risk analysis. Test-driven security risk analysis is a method for carrying out security risk analysis in which security testing is used to support the security risk analysis. The method consists of three main phases. In Phase 1, a security risk analysis is carried out. In Phase 2, security testing is carried out with respect to the security risk analysis. In the 3rd and final phase, the results obtained from the security risk analysis are validated and updated with respect to the test results. 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. To make the evaluation precise, we analysed the difference between the risk model produced before testing and the updated risk model after testing. The results obtained from the case study shows that testing contributes in gaining higher confidence of the correctness of the risk models.

PREPARED BY

Gencer Erdogan

SIGNATURE

**CHECKED BY**

Ketil Stølen

SIGNATURE

**APPROVED BY**

Bjørn Skjellaug

SIGNATURE

**REPORT NO.**

SINTEF A25605

ISBN

978-82-14-05337-1

CLASSIFICATION

Unrestricted

CLASSIFICATION THIS PAGE

Unrestricted

Document history

VERSION	DATE	VERSION DESCRIPTION
1.0	2013-12-03	Final version.
0.2	2013-11-21	Corrected based on first feedback and sent for second quality check.
0.1	2013-10-15	First full draft of the report prior to quality check.

Table of contents

1	Introduction	4
2	Method for Test-driven Security Risk Analysis	5
2.1	Method overview	5
2.2	Description of each step in the method	6
2.2.1	1: Establish context and target of analysis	6
2.2.2	2: Risk identification	8
2.2.3	3: Risk estimation	9
2.2.4	4: Risk evaluation.....	10
2.2.5	5: Test planning	10
2.2.6	6: Test design and implementation	13
2.2.7	7: Test environment set-up and maintenance	13
2.2.8	8: Test execution.....	13
2.2.9	9: Test incident reporting	14
2.2.10	10: Risk validation and treatment	14
3	Research Method	17
4	Overview of the Process Undergone During the Test-driven Security Risk Analysis	18
5	Results: Deviation between risk models before and after testing.....	19
6	Conclusion.....	23

1 Introduction

This report is an evaluation report describing the experiences obtained from a case study, carried out in a period of eight months from June 2012 to January 2013, in which we conducted a test-driven security risk analysis (TSR). Test-driven security risk analysis is a method for carrying out security risk analysis in which security testing is used to support the security risk analysis. Security testing is used to support the security risk analysis as a means for validating and updating the results obtained in the security risk analysis. The customer that commissioned the case study requires full confidentiality. The results that are presented in this report are therefore limited to the experiences from applying the TSR method.

The TSR method consists of three main phases. In Phase 1, a security risk analysis is carried out. In Phase 2, security testing is carried out with respect to the security risk analysis. In the 3rd and final phase, the results obtained from the security risk analysis are validated and updated with respect to the test results. Additionally, risk treatments are also identified as part of the third phase. The security risk analysis was carried out using the CORAS¹ method, while the security testing was carried out following the testing process given by ISO/IEC/IEEE 29119².

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 a change to the risk models.

The test results in this case study led only to a minor change in the risk models. A new vulnerability was detected and we discovered that a certain threat scenario could influence a risk known to us prior to the testing. Furthermore, during the test execution, we confirmed the existence of 7 out of 11 potential vulnerabilities that were identified during the risk analysis. Neither of these findings was sufficiently severe to change the likelihood values in threat scenarios and risks, and thus did not lead to a change in the risk levels of the identified risks. However, the fact that testing did not lead to a major change in the risk models, and consequently no change in the risk levels, is evidence indicating that the risk analysis results obtained prior to the testing were accurate. Testing has thus been used to verify the results obtained during risk analysis and has therefore contributed in gaining higher confidence of the correctness of the risk models.

¹ M.S. Lund, B. Solhaug, and K. Stølen. Model-Driven Risk Analysis: The CORAS Approach. Springer, 2011.

² International Organization for Standardization. ISO/IEC/IEEE 29119-2:2013(E), Software and system engineering - Software testing - Part 2: Test process, 2013.

2 Method for Test-driven Security Risk Analysis

The following sections first give an overview of the main steps of the test-driven security risk analysis method (Section 2.1), followed by a description of each step in the method (Section 2.2).

2.1 Method overview

The steps for the test-driven security risk analysis method are shown in Figure 1.

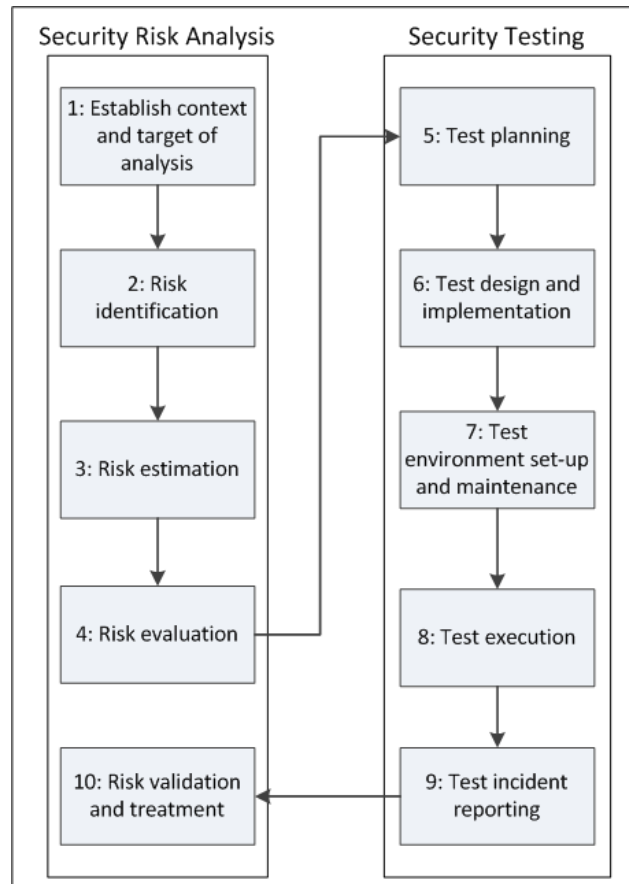


Figure 1 The steps in the test-driven security risk analysis method

Step 1 – 10 is executed in a chronological order, but Step 6 and Step 7 may also be executed in parallel. As Figure 1 shows, the method is divided in two processes addressing security risk analysis and security testing. Step 1, 2, 3, 4 and 10 is executed as part of the security risk analysis process, while Step 5, 6, 7, 8, and 9 is executed as part of the security testing process.

The steps related to the security risk analysis are carried out by making use of the CORAS approach. CORAS is a model-driven approach for carrying out risk analysis and consists of eight steps. The first four steps in CORAS are covered in Step 1 in our method, while the fifth, sixth, seventh and eighth step in CORAS are covered in Step 2, 3, 4 and 10 in our method, respectively. CORAS is also supported by a modelling tool that is available for free at: <http://coras.sourceforge.net/>. This tool has been used during the case study for modelling and documenting the risk analysis, as well as documenting the preparations for prioritizing threat scenarios selected for testing.

The steps related to the security testing are in line with the testing process defined in ISO/IEC/IEEE 29119. The reader is referred to this standard for a description of the testing process. However, for a more thorough description about the concepts used in the context of test-driven risk analysis, the reader is referred to the Conceptual Framework for the DIAMONDS Project³.

2.2 Description of each step in the method

In this section, we describe each of the steps of the method for test-driven security risk analysis in more detail, including an example for each step.

2.2.1 1: Establish context and target of analysis

This step is based on Step 1 – Step 4 of the CORAS risk assessment method. The output of the step is:

- A description of the target of analysis
- A description of the assumptions, focus and scope of the analysis
- CORAS asset diagrams defining assets and parties
- Tables defining consequence and likelihood scales
- Risk matrix tables defining risk evaluation criteria

The description of the target of analysis should be based on the documentation that is already available of the system that is analysed. If this documentation is not sufficient, then a new (high-level) description of the target may have to be specified. A graphical description of the target system (for instance using UML) is preferred as this may make the risk identification easier. The assumptions, focus and scope of the analysis should be document in natural language in addition to the system documentation.

Assets and parties should be documented using CORAS asset diagrams. An asset is something to which a party assigns a value and hence for which the party requires protection. A party is an organisation, company, person, group or other body on whose behalf a risk assessment is conducted. Typically, there is only one party (the customers on whose behalf the risk assessment is conducted), but there may be more than one. Identifying and documenting assets is an important part of the risk assessment as every risk will be related to one or more assets. If a party has no assets to consider, then there is no need to conduct a risk assessment.

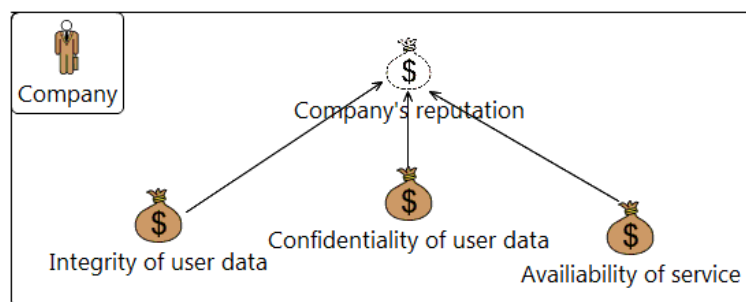


Figure 2 Example of a CORAS asset diagram

An example of a CORAS asset diagram is illustrated in Figure 2. The party (Company) which assigns values to the assets is specified in the top left corner of the diagram. In the diagram proper, three direct assets and one indirect asset are specified. So-called harms relationships between the assets are also specified using arrows. A harms relation expresses that an asset can be harmed through harm to another asset.

³ <http://heim.ifi.uio.no/ketils/kst/Reports/2012.SINTEF-A22798.pdf>

At least one likelihood scale must be defined, and all values of the scale should be precisely defined, typically using intervals of frequencies. An example of a likelihood scale is shown in Table 1.

Table 1 Example of a likelihood scale

Likelihood	Description
Certain	Five times or more per year [50,infinity> : 10y
Likely	Two to five times per year [20,50> : 10y
Possible	One to two times per year [10,20> : 10y
Unlikely	One to ten times per ten years [1,10> : 10y
Rare	Less than once per ten years [0,1> : 10y

In addition, one consequence scale for each asset should be defined. An example of the definition of consequence scales for the assets "Availability of service" and "Confidentiality of user data" is shown in Table 2 and Table 3, respectively.

Table 2 Example of a consequence scale for asset Availability of service

Consequence	Description
Catastrophic	Downtime within interval [6 hours, ...>
Major	Downtime within interval [60 min, 6 hours>
Moderate	Downtime within interval [30 min, 60 min>
Minor	Downtime within interval [5 min, 30 min>
Insignificant	Downtime within interval [0 min, 5 min>

Table 3 Example of a consequence scale for asset Confidentiality of user data

Consequence	Description
Catastrophic	[50, ...> users are affected
Major	[5, 50> users are affected
Moderate	[2, 5> users are affected
Minor	[1, 2> users are affected
Insignificant	[0,1> users are affected

Having defined likelihood and consequence scales, risk evaluation criteria should be defined using risk matrices. It is easiest to define only one risk evaluation matrix. However, sometimes it makes more sense to define one risk matrix per asset.

An example of a risk evaluation matrix is given in Table 4. Here, risk values are denoted by green, yellow, or red. It's up to the risk analyst(s) to define what is meant by these, but typically risks that have a red risk value are unacceptable and must be treated, risks that are yellow must be monitored, and risks that are green are acceptable.

Table 4 Example of risk evaluation criteria

		Likelihood				
		Rare	Unlikely	Possible	Likely	Certain
Consequence	Insignificant					
	Minor					
	Moderate					
	Major					
	Catastrophic					

2.2.2 2: Risk identification

This step corresponds to Step 5 of the CORAS method. The objective is to identify unwanted incidents, threats, threat scenarios and vulnerabilities w.r.t. the assets that were identified. The output of this activity is:

- A set of CORAS threat diagrams.

Risks should be identified in workshops where risks and their causes are systematically identified by workshop participants. The description of the target of evaluation should be used as a basis for the risk identification and CORAS threat diagrams should be used to document the results on the fly during the risk identification process.

CORAS threat diagrams specify directed acyclic graphs in which the nodes are of one the following kinds:

- **Threat:** A potential cause of an unwanted incident (illustrated by a "man-icon" with a warning sign in case of a human threat).
- **Threat scenario:** A chain or series of events that is initiated by a threat and that may lead to an unwanted incident (illustrated by ellipses with warning signs).
- **Unwanted incident:** An event that harms or reduces the value of an asset (illustrated by box with a star in the top right corner).
- **Asset:** Something to which a party assigns value and hence for which the party requires protection (illustrated by money bags).

Relations may be of one of the following kinds:

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

In addition, relations may be annotated by a

- **Vulnerability:** A weakness, flaw or deficiency that opens for, or may be exploited by, a threat to cause harm to or reduce the value of an asset (illustrated by red open locks).

An example of a CORAS threat diagram is shown in Figure 3.

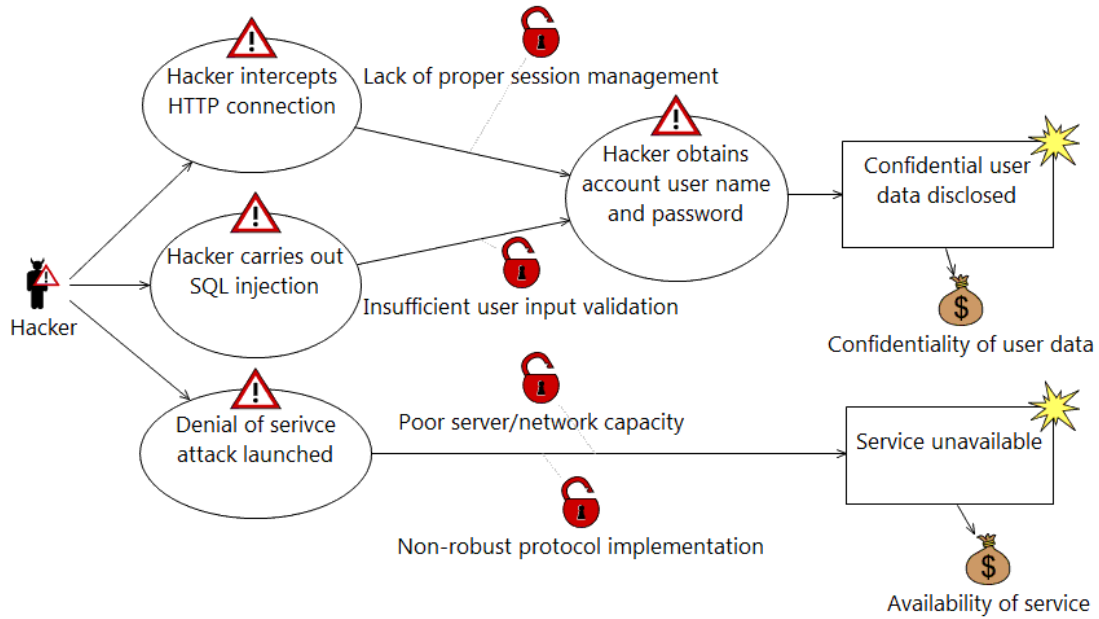


Figure 3 Example of a CORAS threat diagram

2.2.3 3: Risk estimation

This step corresponds to step 6 of the CORAS method. The objective is to identify (1) likelihood values for threat scenarios and unwanted incidents, (2) conditional likelihoods between threat scenarios, and threat scenarios and unwanted incidents, (3) consequence values on relations between unwanted incidents and assets. The output of the activity is:

- A set of CORAS threat diagrams with likelihood and consequence values.

Similar to the risk identification activity, risk estimation should be done in workshops where the workshop participants estimate likelihood values based on expert judgment, historical data, a combination of both, etc.

An example of a CORAS threat diagram with likelihood and consequence values is given in Figure 4. Note that the likelihood values (such as Possible) and the consequence values (such as Major) that are used in the diagram should be taken from the likelihood and consequence scales defined in Step 1.

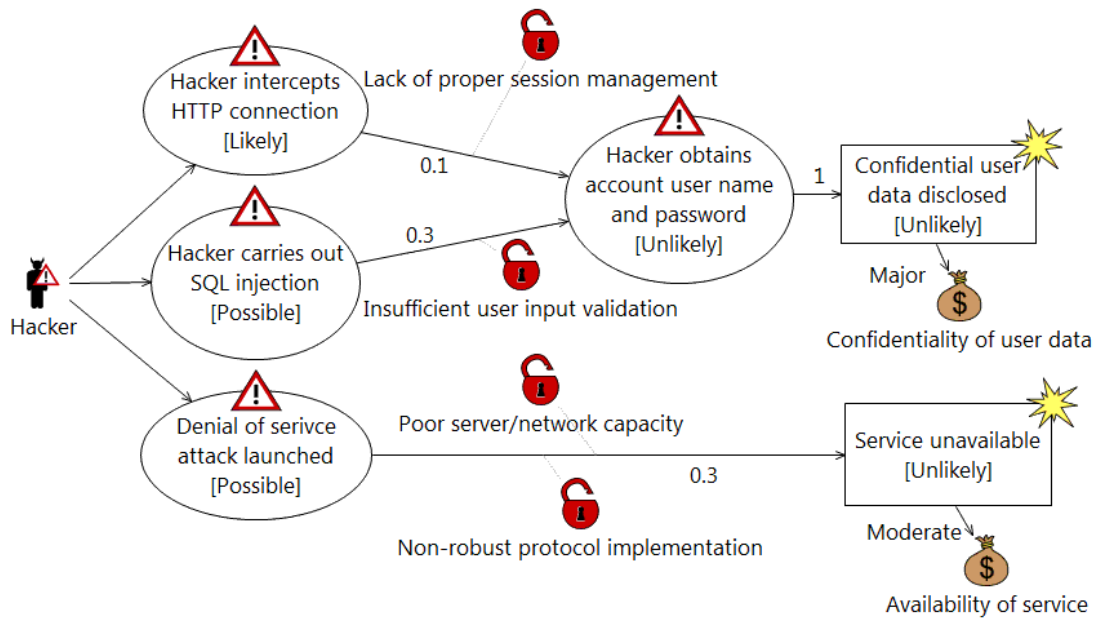


Figure 4 Example of a CORAS threat diagram with likelihood and consequence values

2.2.4 4: Risk evaluation

This step corresponds to Step 7 of the CORAS method. The objective is to identify and priori-tize/evaluate risks, where a risk is understood as an unwanted incident together with a likelihood and a consequence value. The risk value of a risk is obtained by plotting the risk into a risk matrix (defined in Step 1).

Table 5 shows an example of a risk evaluation matrix where the two risks shown in Figure 4 have been inserted (a risk can be seen as an unwanted incident together with a likelihood value and a consequence value). Here we see that the risk *Service unavailable* has risk value green, while the risk *Confidential user data disclosed* has risk value yellow.

Table 5 Example of a risk evaluation matrix with risks

		Likelihood				
		Rare	Unlikely	Possible	Likely	Certain
Consequence	Insignificant	Green	Green	Green	Yellow	Yellow
	Minor	Green	Green	Green	Yellow	Red
	Moderate	Green	Service unavailable	Yellow	Red	Red
	Major	Yellow	Confidential user data disclosed	Red	Red	Red
	Catastrophic	Yellow	Red	Red	Red	Red

2.2.5 5: Test planning

The objective of the test planning step is to identify, prioritize and select threat scenarios in the CORAS threat diagrams that have been specified up to this point. Threat scenarios are used for identifying possible test scenarios because they say something about how to test. They typically describe some sort of misuse-case or attack, and this can be used as a basis for security testing. The output of the activity is:

- A table describing all prioritized threat scenarios in the CORAS risk diagrams, with an indication of the selected threat scenarios for testing.

By prioritizing the threat scenarios we also prioritize the test scenarios they represent. The prioritization of threat scenarios is calculated based on three values:

- **S (Severity):** An estimate of the impact that the threat scenario has on the risks. The severity estimate is assigned per threat scenario by taking the threat scenario's likelihood value *and* the conditional probability value on the edge(s) going out from the threat scenario. The higher the severity value in a threat scenario is the higher reason there is to test the scenario it represents.
- **T (Testability):** An estimate of how testable a threat scenario is. Typically, testability is an estimate of the effort required to implement the concrete test cases of the scenario given the tools and expertise available, but it can also be based on other considerations such as ethics. For instance, to test scenarios related to "Social engineering", one might have to "fool" employees of an organization, and this might not be considered ethical. The higher the testability, the higher the priority should be.
- **U (Uncertainty):** An estimate of how uncertain one is about the correctness of the likelihood value of a threat scenario. If the uncertainty is very low, then there might not be a need for testing, since then the test results may not give any new information. Conversely, a high uncertainty suggests that the test scenario should be prioritized for testing.

The severity, testability and uncertainty values must be supplied by the user. Table 6, Table 7, and Table 8, shows the scales used to assign values for severity, testability and uncertainty, respectively.

Table 6 Severity scale

	Very low severity	Low severity	Medium severity	High severity	Very high severity
Value	2-3	4-5	6-7	8-9	10

Table 7 Testability scale

	Very low testability	Low testability	Medium testability	High testability	Very high testability
Value	1	2	3	4	5

Table 8 Uncertainty scale

	No uncertainty	Very low uncertainty	Low uncertainty	Medium uncertainty	High uncertainty	Very high uncertainty
Value	0	1	2	3	4	5

Figure 5 shows an example of a CORAS threat diagram where all threat scenarios are annotated with labels of the form S:X, T:Y, U:Z, specifying that the corresponding threat scenario has a severity value X, testability value Y and an uncertainty value Z.

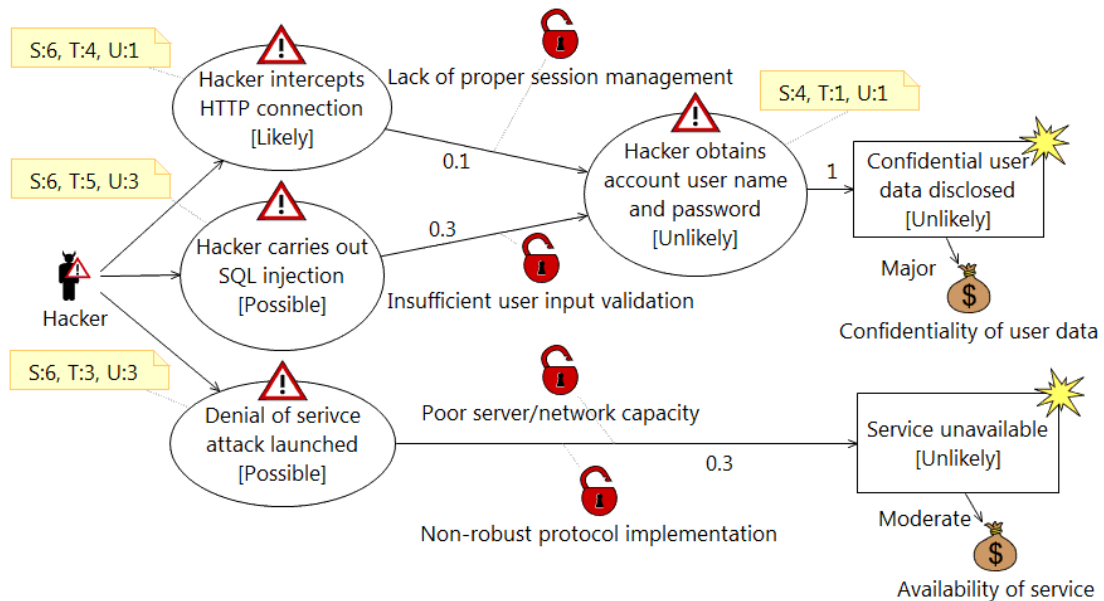


Figure 5 Example of a CORAS threat diagram with annotations for severity, testability and uncertainty

Having annotated the risk model with severity, testability and uncertainty values, a table containing a prioritized list of threat scenarios can be generated. An example of such a table is shown in Table 9. Here all test scenarios correspond to threat scenarios in Figure 5, and values for severity (S), testability (T) and uncertainty (U) have been taken from the annotation in the diagram. Based on the values for severity, testability, and uncertainty, a priority value can be automatically calculated. In the example, we have taken the priority of a test scenario to be the sum of S, T, and U. However, other functions are possible.

$$priority(TS) = scale(S) + T + U$$

Where TS is the threat scenario being prioritized and where the scale function is defined as

$$scale(S) = \left(\left(\frac{S - 2}{8} \right) \times 4 \right) + 1$$

Based on the list of prioritized threat scenarios, the user must select the threat scenarios that should be refined into executable test cases. For instance, the user can do this by setting a priority threshold PT, and then select all threat scenarios that have a higher priority than PT. In Table 9, the three first threat scenarios with the highest priority values are shown in boldface, indicated that these three threat scenarios are selected for further testing, and the others not.

Table 9 Example of a prioritized list of threat scenarios

Id	Threat scenario (test scenario)	S	T	U	Priority
TS2	Hacker carries out SQL injection	6	5	3	11
TS3	Denial of service attack launched	6	3	3	9
TS1	Hacker intercepts HTTP connection	6	4	1	8
TS4	Hacker obtains account user name and password	4	1	1	4

2.2.6 6: Test design and implementation

The threat scenarios identified and selected in Step 5 may be tested in various ways. The purpose of this step is to analyse the threat scenarios and specify these into one or more security test cases. This is something the analyst has to carry out. The following points describe one way of specifying security tests from threat scenarios:

1. Analyse the description of the test scenario and use resources, such as OWASP and CVE, to specify one or more test cases.
 2. Document the following for each test case:
 - a. A unique id in the following format: TSX-Y where X is the id of the threat scenario being tested, and where Y is a unique number for one test case addressing the threat scenario.
 - b. Describe the test objective.
 - c. Document the following preconditions for the test case:
 - i. Threat profile (what kind of threat profile may initiate the threat scenario represented by the test case?)
 - ii. Technicality (what kind of technical preconditions has to be considered and has to be in place in order to carry out the test case?)
 - d. Document the test procedure
 - i. Tools (what tools are to be used to carry out the test case?)
 - ii. Procedure (how shall the test case be executed? – It is not always possible to document the test procedure in detail before carrying out the test case, e.g., when carrying out exploratory testing. The procedure should therefore be updated and properly documented during/after the test case has been executed so that it may be re-executed in a possible re-test).
 - e. Document the vulnerability/vulnerabilities addressed by the test case.
 - f. Document the verdict of the test case (in what situation has the test case passed, failed or is inconclusive? – additional verdicts may be used).
- The output of the activity is documentation describing test cases with respect to the points above.

2.2.7 7: Test environment set-up and maintenance

The objective of this step is to set up the test environment (and maintain it) in which the test cases are to be executed. This is an optional step; if the test environment is already set up then this can be skipped, otherwise it has to be set up. Our method does not explicitly explain the manner in which the test environment is set up and maintained. We refer to the guidelines given by ISO/IEC/IEEE 29119 for setting up the test environment and maintaining it.

2.2.8 8: Test execution

The objective of Step 8 is to; (1) execute the test cases specified in Step 6 on the test environment specified in Step 7, (2) note any divergence in the execution of the test cases from the procedures described in Step 6, (3) note a verdict for the test cases by comparing expected results with actual results, (4) note the vulnerabilities, and their severity, addressed by the test case. The justification of the vulnerabilities' severity level has to be documented during this process. Table 10 shows a generic table for documenting the test results. The output of this step is:

- A table documenting the test results including the severity of the vulnerability (or vulnerabilities) addressed by the test case.
- Documented justification of the vulnerabilities' severity level.

Table 10 Example of a generic test result report

Test	Verdict	Verdict explanation	Vulnerability	Severity of vulnerability
TS1-1	Pass / Fail / Inconclusive	Explain based on what the verdict is set.	Description of vulnerability	A description of the severity, e.g., in form of High / Medium / Low
TS1-2
TS2-1
Etc.

2.2.9 9: Test incident reporting

The objective of Step 9 is to report on the test results, and especially highlighting the tests that have verdicts other than pass. This is of particular importance in the final step, where the test case results are used as a basis to validate the risk analysis. The output of this step is:

- A table giving an overview of all the vulnerabilities addressed during the test process. Table 11 shows an example of a generic table for documenting the overview of all vulnerabilities addressed.
- A table giving an overview of the test results. Table 12 shows an example of a generic table for documenting the overview of the test results.

Table 11 Example of a generic vulnerability overview table

Vulnerability	Description
V1	Give a short description of the vulnerability.
V2	.
Etc.	.

Table 12 Example of a generic test results overview table

Test	Description	Verdict	Vulnerability	Severity of vulnerability
TS1-1	Short description of test case.	Pass / Fail / Inconclusive	V1	A description of the severity, e.g., in form of High / Medium / Low
TS1-2
TS2-1
Etc.

2.2.10 10: Risk validation and treatment

The purpose of Step 10 is to (1) validate the risk model based on the test results obtained in Step 9 and to update the risk model based on the test results if necessary, (2) propose treatments for the most severe risks. The output of the activity is:

- An updated CORAS risk model
- An updated Risk evaluation matrix
- A CORAS treatment diagram

The user should use the test report as a basis for validating and updating the CORAS risk model. Specifically, by examining each test case result for a given threat scenario, the user must decide whether the overall likelihood value of the threat scenario should be increased or decreased. For example, assume there are two test cases for threat scenario TS2 (see Table 9); TS2-1 and TS2-2. TS2-1 and TS2-2 represent two separate test cases for SQL injection. Let's say that both of these test cases have yielded the verdict *Fail* by exploiting the vulnerability *Insufficient user input validation*. Note that *Fail* means that TS2-1 and TS2-2 were successfully carried out. Furthermore, we assume that the vulnerability has a high severity on the system being tested. Taking all this into consideration we increase the likelihood of threat scenario *Hacker carries out SQL injection* from *Possible* to *Likely*. The level of increase is something the analyst has to decide.

Figure 6 shows that the likelihood in threat scenario *Hacker carries out SQL injection* has been updated from *Possible* to *Likely*. By updating this, we see that it has an effect on the threat scenario it leads to and consequently the unwanted incident it leads to. Thus, the likelihood value in threat scenario *Hacker obtains account user name and password* and unwanted incident *Confidential user data disclosed* is updated from *Unlikely* to *Possible* due to the likelihood increase in threat scenario *Hacker carries out SQL injection*.

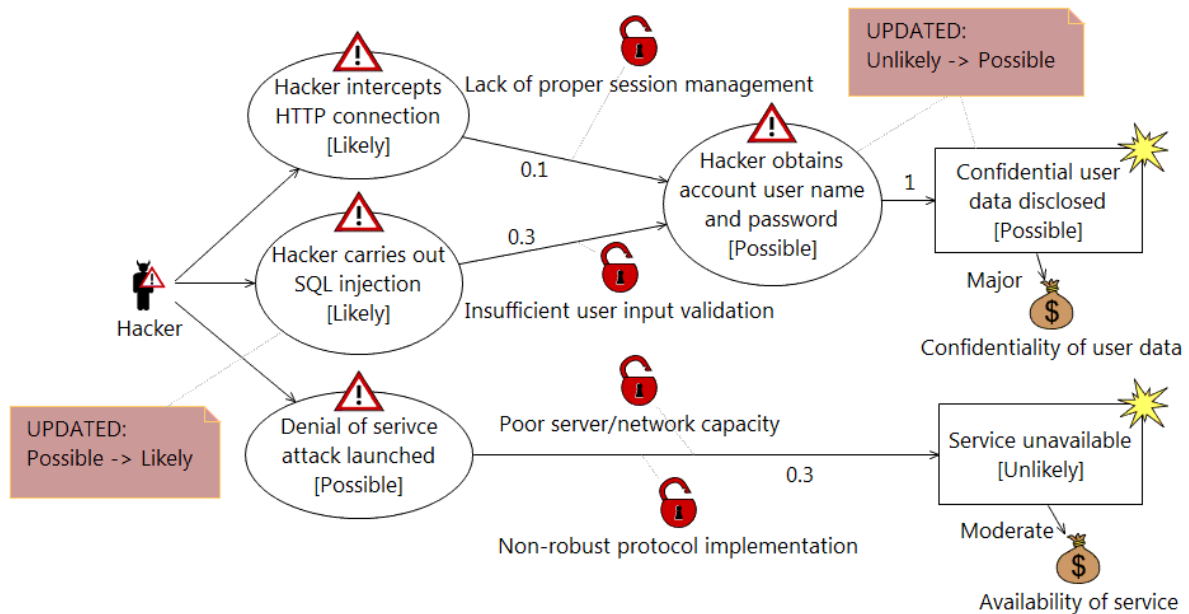


Figure 6 Validated and updated risk model based on the test results

If the update of the risk model resulted in changes to the likelihood values of the unwanted incidents in the model, then the risk evaluation matrix must also be updated. For instance, since the likelihood value of the unwanted incident *Confidential user data disclosed* has changed from *Unlikely* to *Possible* in Figure 6, then the corresponding risk is categorized as red when plotting it into the risk evaluation table (see Table 13). Comparing the previous risk evaluation matrix (Table 5) to the updated matrix, we see that one risk has changed from yellow to red as a result of taking the test results into account.

Table 13 Example of an updated risk evaluation matrix

		Likelihood				
		Rare	Unlikely	Possible	Likely	Certain
Consequence	Insignificant					
	Minor					
	Moderate		Service unavailable			
	Major			Confidential user data disclosed		
	Catastrophic					

After the risk model and the risk evaluation matrix has been updated based on the test results, treatments for the most severe risks must be proposed. This should be done according to the CORAS methodology, where treatments are specified in so-called CORAS treatment diagrams.

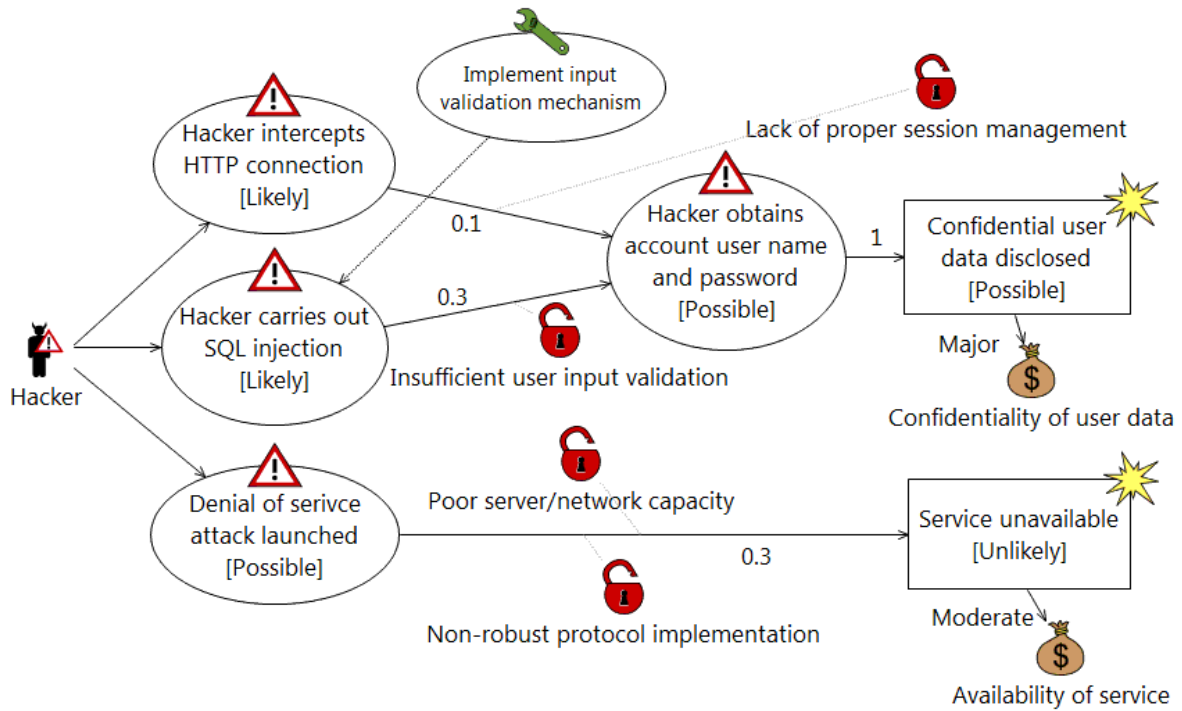


Figure 7 Example of a CORAS treatment diagram

3 Research Method

The basis of our evaluation is to compare the risk models produced before and after testing. By risk models, we mean CORAS threat diagrams as described throughout Section 2.2. A CORAS threat diagram is set up of risk model elements that may be grouped in three categories; nodes, relations or assignments.

As illustrated by the examples in Section 2.2, a CORAS threat diagram 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.

A relation in a CORAS threat 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 initiates relations and the leads to relations going from A to B, describing a weakness, flaw or deficiency that opens for A leading to B.

Two CORAS risk models are equal if they contain exactly 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 is different in RM_B and RM_A .

4 Overview of the Process Undergone During the Test-driven Security Risk Analysis

The case study was carried out through 7 workshops. An overview of these workshops is given in Table 14. In the participants column, the denotations C:*n* and A:*m* represent *n* participants from the customer side and *m* participants from the risk analysis team.

Table 14 Overview of the workshops

Date	Step	Participants	Duration
19.06.12	Step 1: Establish context and target of evaluation.	C:7, A:4	5 hours
23.08.12	Completion of Step 1. Step 2: Risk identification.	C:5, A:3	6 hours
21.09.12	Completion of Step 2. Step 3: Risk estimation.	C:4, A:3	6 hours
01.10.12	Step 4: Risk evaluation. Assignment of Severity, Testability and Uncertainty on threat scenarios.	C:3, A:3	2.5 hours
16.11.12	Step 5 and Step 6: Test planning and test design were carried out before the meeting. The meeting was used to go through the test cases from Step 6, and also agree upon an overall test plan. Step 7 was not necessary to carry out as the test environment was already set up.	C:6, A:3	3 hours
18.12.12	Step 8: Test execution was carried out for 17 hours during three working days. Step 9: Test incident reporting was carried out based on the test results, after test execution was completed.	C:4, A:3	17 hours
19.12.12			
20.12.12			
24.01.13	Step 10: Risk validation and treatment was carried out before the meeting and the documentation was sent to the customer. The meeting was used to discuss the results.	C:4, A:3	2 hours

The steps in the method were carried out as described in Section 2.2. The test execution process was carried out automatically, semi-automatically, and manually:

- Automated testing: IBM Rational Software Architect, Smartesting CertifyIt, In-house developed software for automating the browser, OWASP Web Scarab and Burp Suite Free Edition (for automatic analysis purposes).
- Semi-automated testing: OWASP Web Scarab.
- Manual testing: Google Chrome.

The automated process was as follows. First we specified a model of the system under test using IBM RSA together with the CertifyIt plugin, then we used CertifyIt to generate abstract Java tests, then we concretized these in Java, and finally we executed them using our software for automating the browser. OWASP Web Scarab was used to automatically execute some of the tests and Burp Suite was used to automatically analyse the test results obtained from OWASP Web Scarab. 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 analysed.

5 Results: Deviation between risk models before and after testing

In this section we first give an overview of the differences between the risk models before and after testing, followed by a more detailed description of the differences.

Figure 8 shows a chart that gives an overview of the number of nodes in the risk model before and after testing, as well as the number of vulnerabilities before and after testing. Figure 9 shows a chart that gives an overview of the number of relations in the risk model before and after testing.

We see from the chart in Figure 8 that no nodes were added or deleted after testing. Only a vulnerability assignment was added to the risk model after testing. Furthermore, we see from the chart in Figure 9 that one leads to relation was added to the risk model after testing and that no relations were deleted 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 vulnerability that could be exploited by the attack described by the threat scenario. The chart in Figure 10 gives an overview of the total number of threat scenarios, risks and vulnerabilities that were identified during the risk analysis. All of the vulnerabilities that were identified during risk analysis were categorized as potential vulnerabilities, because it was not clear whether they existed or not. However, this is not necessarily the case for all risk analyses. Vulnerabilities that are identified during a risk analysis may well be known to exist in a target of evaluation. Furthermore, the chart shows that 8 threat scenarios (13% of the total threat scenarios) were tested and that by testing these 8 threat scenarios 13 threat scenarios (33% of the total threat scenarios), 14 risks (54% of the total risks), and 11 vulnerabilities (85% of the total potential vulnerabilities) were potentially affected by testing. By potentially affected, we mean the three kinds of changes (add, delete or edit) testing may cause on a risk model. These three changes are done by analysing the test results. For example, let us say that we test a threat scenario TS and that the test result provides evidence to increase the likelihood value of TS. Increasing the likelihood value of TS may in turn lead to an increase of the likelihood values of all threat scenarios or risks caused by TS.

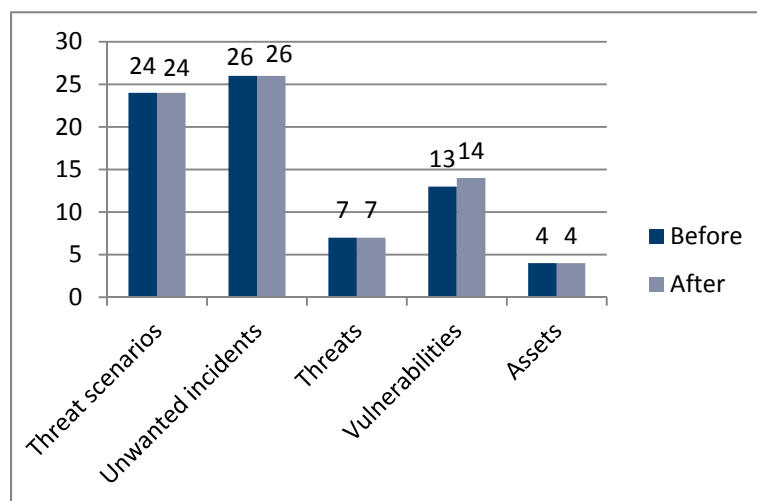


Figure 8 Number of nodes and vulnerabilities, before and after testing

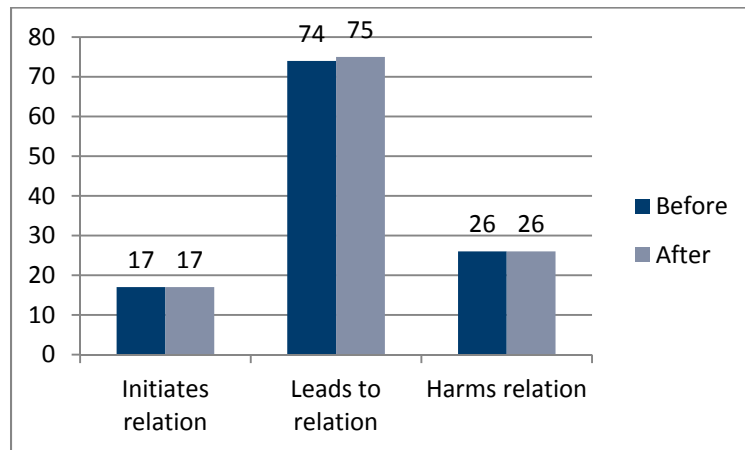


Figure 9 Number of relations, before and after testing

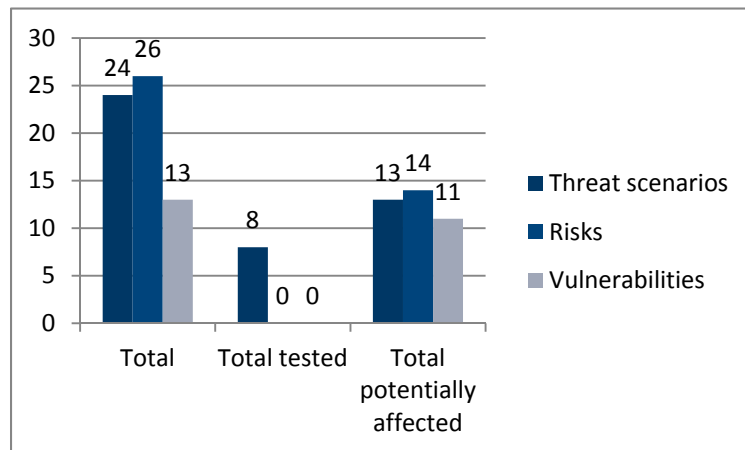


Figure 10 Number of threat scenarios tested and the number of threat scenarios, risks and vulnerabilities potentially affected by the testing

Figure 11 shows an excerpt of the risk model from the case study, representing the threat scenarios selected for testing and the risk model elements potentially affected by testing. The dashed ovals represent the threat scenarios that were selected for testing. The solid ovals and rectangles represent the threat scenarios and risks (unwanted incidents), respectively, potentially affected by testing. The dashed pentagons represent potential vulnerabilities identified during the risk analysis. Note that the threat scenarios selected for testing are also considered as risk model elements that may potentially be affected by testing. The triangles represent threats.

The numerical values in the threat scenarios and the risks represent a likelihood value in a scale from 1 to 5, where 1 represent the lowest likelihood value and 5 represent the highest likelihood value. With respect to the potential vulnerabilities, testing may help us clarify the uncertainty of whether vulnerabilities are potential or not. A potential vulnerability detected by testing is no longer a potential vulnerability. However, a potential vulnerability not detected by testing is still a potential vulnerability as testing may never provide information about the absence of a vulnerability. In that case we are more *certain* that the vulnerability does not exist.

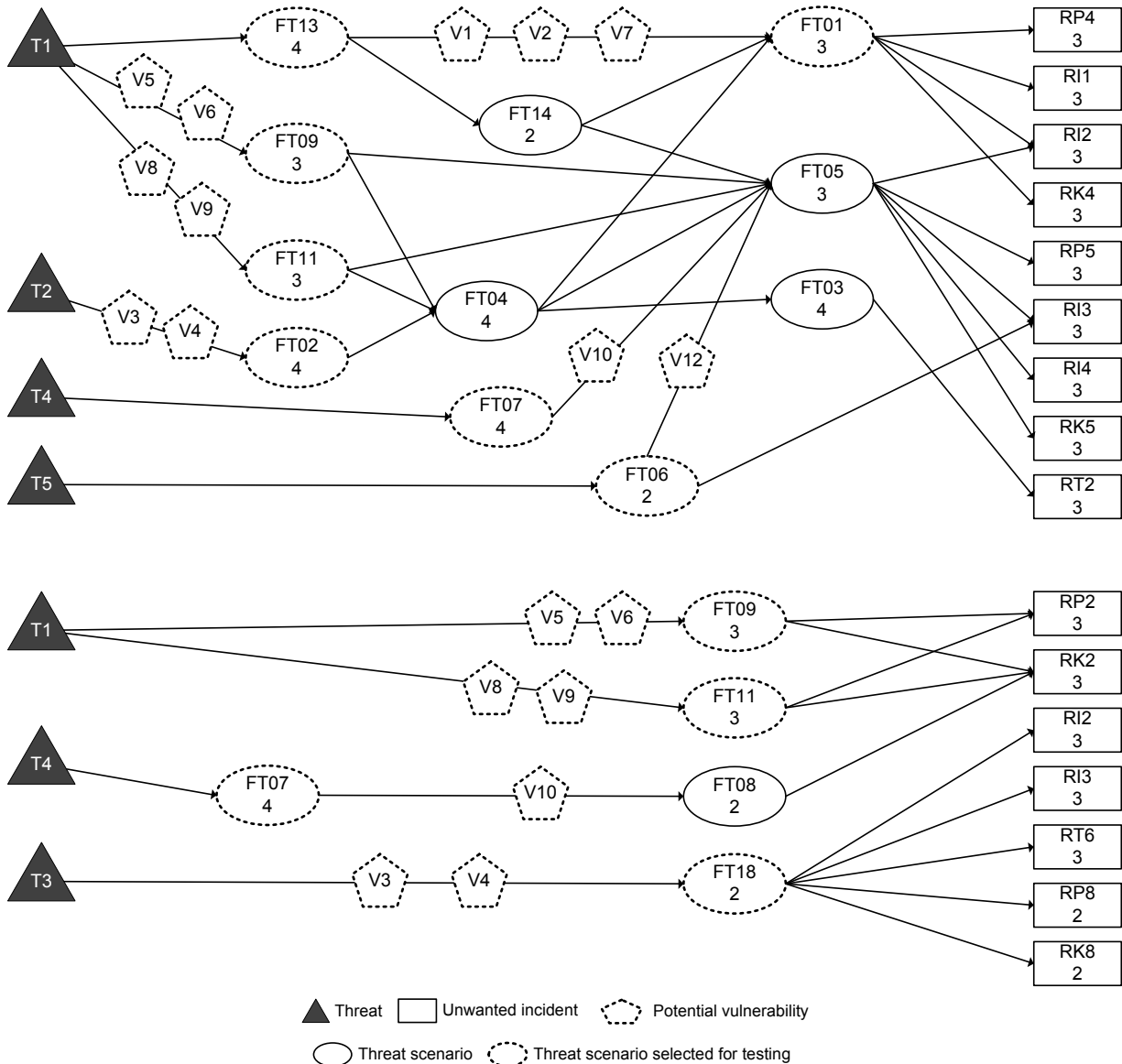


Figure 11 Risk model before testing

Figure 12 shows the difference between risk model elements before and after 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 following results can be observed from the threat model in Figure 12:

- None of the tested threat scenarios changed likelihood value after testing, hence, none of the threat scenarios and risks potentially affected by the testing changed likelihood value.
- The threat scenarios that were directly tested detected 7 out of 11 potential vulnerabilities (represented by a solid pentagon). The remaining 4 potential vulnerabilities are still potential vulnerabilities after testing.
- By testing threat scenario FT06 we discovered that FT06 also may lead to risk RT2 by exploiting a previously unknown vulnerability V11.
- The leads to relation from FT06 to RT2, and the vulnerability assigned on this relation are risk model elements that were added to the risk model due to information obtained from testing.

Although 7 vulnerabilities were detected by testing the threat scenarios, they were not sufficiently severe to affect the likelihood of the tested threat scenarios.

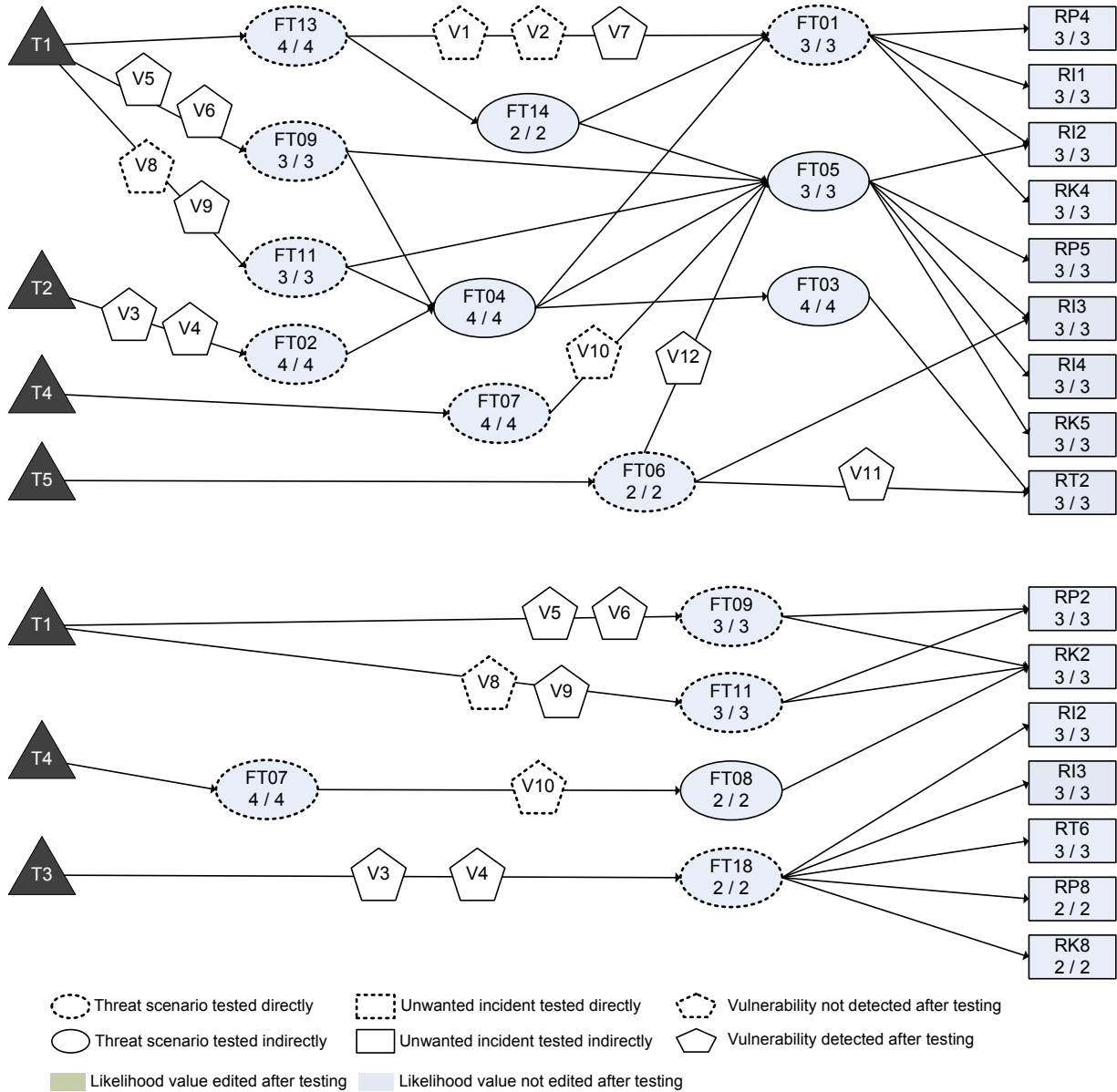


Figure 12 Difference between risk models before and after testing

6 Conclusion

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 analysed the difference between the risk model produced before testing and the risk model produced after testing.

The process of testing yielded information which led only to a minor change in the risk model created before testing. 7 out of 11 potential vulnerabilities were confirmed and thereby changed from potential vulnerabilities to detected vulnerabilities. A new vulnerability and one new leads-to relation was added to the risk model. It is worth noting that this one vulnerability was uncovered due to test execution and would never have been uncovered in the risk analysis phase, regardless of how much effort we would have spent. This minor change did not lead to any change of the risk levels and did not contribute in identifying new risks. However, the fact that testing did not lead to a major change in the risk models, and consequently no change in the risk levels, is evidence indicating that the risk analysis results obtained prior to the testing were accurate. Testing has thus verified the results obtained during risk analysis and has therefore contributed in gaining higher confidence of the correctness of the risk models.



Technology for a better society

www.sintef.no