



Title: DIAMONDS Security Testing Methodology

Version: 1.0

Date : 16.05.2013

Pages : 39

Editor: Fredrik Seehusen

Reviewers: Bruno Legeard, Peter Schmitting

To: DIAMONDS Consortium

The DIAMONDS Consortium consists of:

Codenomicon, Conformiq, Dornier Consulting, Ericsson, Fraunhofer FOKUS, FSCOM, Gemalto, Get IT, Giesecke & Devrient, Grenoble INP,itrust, Metso, Montimage, Accurate Equity, SINTEF, Smartesting, Secure Business Applications, Testing Technologies, Thales, TU Graz, University Oulu, VTT

Status:

[] Draft
[] To be reviewed
[] Proposal
[X] Final / Released

Confidentiality:

[X] Public Intended for public use
[] Restricted Intended for DIAMONDS consortium only
[] Confidential Intended for individual partner only

Deliverable ID: D5_4_T1-T3

Title:

DIAMONDS Security Testing Methodology

Contributors: Fredrik Seehusen, Johannes Viehmann, Stephane Maag, Jürgen Großmann



	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 2 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

TABLE OF CONTENTS

1. Overview of related processes and standards.....	6
1.1 Testing	6
1.2 Security Testing	8
1.3 Model-based testing.....	9
1.4 Risk Assessment	10
2. A generic methodology for model-based security testing.....	12
2.1 A generic process for model-based security testing	12
2.2 Relating DIAMONDS techniques to the methodology	14
3. A specific methodology for model-based security testing	16
3.1 Overview: combined TMSR and RMST	16
3.2 From risk analysis artefacts to test patterns	17
3.2.1 Selecting elements to test.....	17
3.2.2 Map security test patterns to threat scenarios	18
3.3 From test patterns to Test implementation and execution	18
3.4 Complete the Iteration with TMSR	19
4. A specific methodology for test-based risk assessment.....	20
4.1 Overview of process and instantiation	20
4.1.1 Overview of process	20
4.1.2 Instantiation	21
4.2 Description of the process	22
4.2.1 1: Establish context and target of analysis.	22
4.2.2 2: Risk identification	24
4.2.3 3: Risk estimation	25
4.2.4 4: Risk evaluation	26
4.2.5 5.a: Test identification.....	26
4.2.6 5.b: Test selection/prioritization	27
4.2.7 6: Test design, implementation, and execution	29
4.2.8 7: Risk validation and treatment	30
5. A Catalogue of Security Test Patterns	33
5.1 Overview on the DIAMONDS Security test patterns.....	33
5.2 Security Test Pattern Application in the Case Studies.....	35
5.2.1 Banking case study: Giesecke & Devrient (FOKUS).....	35
5.2.2 Automotive case study: Dornier Consulting (IT)	36
5.2.3 Smart cards case study: Gemalto (INPG)	37
6. Conclusion.....	38
7. References	39


	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 3 of 39
		Version: 1.0
		Date : 16.05.2013
		Status : Final Confid : Public

FIGURES

Figure 1 Activities of a three layered test process [8].....	6
Figure 2 A generic testing process.....	7
Figure 3 Four-stage penetration testing methodology	9
Figure 4 A generic model-based testing process.....	9
Figure 5 A generic risk management process.....	10
Figure 6 The generic process for model-based security testing and its relation to the testing process.....	13
Figure 7 Relating DIAMONDS techniques to the generic process.....	15
Figure 8 Model-based security testing process.....	17
Figure 9: Mapping threat diagram artefacts to test pattern	19
Figure 10 A specific test-based risk assessment process.....	20
Figure 11 Relation the specific test-based risk assessment process to the generic process	22
Figure 12 Example of a CORAS asset diagram.....	23
Figure 13 Example of a CORAS threat diagram	25
Figure 14 Example of a CORAS threat diagram with likelihood values	26
Figure 15 Example of a CORAS threat diagram with annotations for testability and uncertainty	28
Figure 16 Example of an updated risk model based on test results	31
Figure 17 Example of a CORAS treatment diagram	32
Figure 18: Application of test pattern to risk assessment artefacts	36

TABLES

Table 1 Example of a likelihood scale	23
Table 2 Example of a consequence scale for asset Availability of service	23
Table 3 Example of a consequence scale for asset Confidentiality of user data	24
Table 4 Example of risk evaluation criteria.....	24
Table 5 Example of a risk evaluation matrix with risks.....	26
Table 6 Example of identified test scenarios.....	27
Table 7 Example of a prioritized list of test scenarios	29
Table 8 Example of test result report	30
Table 9 Example of an updated risk evaluation matrix	31


	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 4 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

HISTORY

Vers.	Date	Author	Description
0.1	05/03/13	Fredrik Seehusen	Template created
0.2	05/04/13	Stephan Maag	Draft input on test pattern catalogue provided
0.3	12/04/13	Fredrik Seehusen	Draft input on process overview, generic process description, and test-based risk assessment process provided.
0.4	15/04/13	Jürgen Großmann	Input on test pattern catalogue
0.5	29/04/13	Fredrik Seehusen	Process overview, generic process description, and a process for test-based risk assessment ready for internal review.
0.9	07/05/13	Johannes Viehmann	Input on a specific methodology for model-based security testing
1.0	16/05/13	Fredrik Seehusen	Finalized document based on feedback from reviewers

APPLICABLE DOCUMENT LIST

Ref.	Title, author, source, date, status	DIAMONDS ID

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 5 of 39
		Version: 1.0
		Date : 16.05.2013
		Status : Final Confid : Public

EXECUTIVE SUMMARY

This document constitutes the third and final deliverable of work package 4, documenting results of task T4.1 (security patterns) and tasks 4.2 and task 4.3 on risk- and model-based security testing methodologies. While the other work packages of the DIAMONDS project describe techniques/methods and tools, work package 4 describes processes/guidelines for applying these tool and techniques in practice.

The main objectives of the deliverable is to document a generic process for model-based security testing, provide examples of a concrete instances of the process, and to provide an overview of the security test patterns that have been identified in the DIAMONDS project.

The deliverable is structured as follows. First, in Section 1, we give an overview of existing processes within the main areas addressed by the DIAMONDS project (testing, security testing, model-based testing, and risk assessment). Then in, Section 2, we present a generic process for model-based security testing which combines the processes presented in Section 1, and give an overview of how the DIAMONDS techniques relate to the activities of the process. In Section 3 and Section 4, we present concrete instances/refinements of the generic process, focusing on model-based security testing and test-based security risk assessment, respectively. Finally, in Section 5, we provide an overview of the security test patterns that have been identified in the DIAMONDS project, and describe their application in the DIAMONDS case studies.

1. OVERVIEW OF RELATED PROCESSES AND STANDARDS

In this section we describe processes and standards that we have used as a basis for defining the generic DIAMONDS process for model-based security testing (described in Section 2). For each of the main areas addressed by DIMAONDS (testing, security testing, model-based testing, and risk assessment), we define a process which is representative for that area. In Section 2, we combine all these processes into what we call the generic DIAMONDS process for model-based security testing.

1.1 TESTING

In this section, we define a generic process for testing based in the upcoming standard ISO/IEC 29119 Software Testing [8]. According to [8], the testing activities that are performed during the life cycle of a software system may be grouped into a three layered test process, as shown in Figure 1.

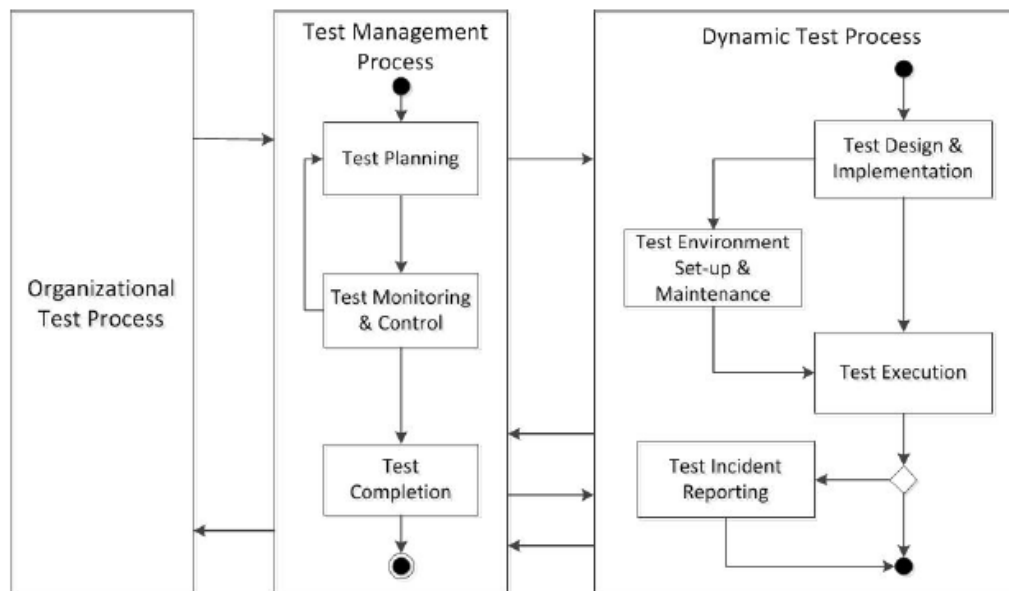


Figure 1 Activities of a three layered test process [8]

The aim of the organizational test process layer is to define a process for the creation and maintenance of organizational test specifications, such as organizational test policies, strategies, processes, procedures and other assets [8].

The aim of the test management process layer is to define processes that cover the management of testing for a whole test project or any test phase or test type within a test project (e.g. project test management, system test management, performance test management) [8].

The aim of the dynamic test process layer is to define generic processes for performing dynamic testing. Dynamic testing may be performed at a particular phase of testing (e.g. unit, integration, system, and acceptance) or for a particular type of testing (e.g. performance testing, security testing, and functional testing) within a test project [8].

What we refer to as the testing process in the DIAMONDS project is basically what Figure 1 refers to as the dynamic test process. However, as indicated by Figure 2 we also add a test planning step capturing the test planning of the test management process of relevance for one run of the dynamic test process.

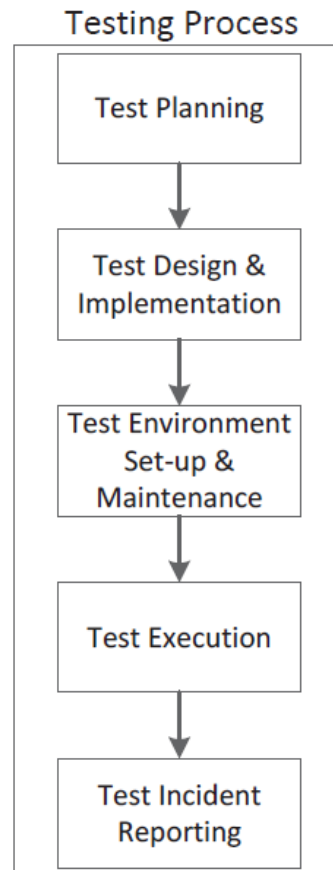


Figure 2 A generic testing process

Below, we define the activities of the generic testing process in more detail.

Test Planning

The test planning is the process of developing the test plan. Depending on where in the project this process is implemented this may be a project test plan or a test plan for a specific phase, such as a system test plan, or a test plan for a specific type of testing, such as a performance test plan (adapted from [8]).

Test Design and Implementation

The test design and implementation is the process of deriving the test cases and test procedures (adapted from [8]).


Test Environment Set-up and Maintenance

The test environment set-up and maintenance process is the process of establishing and maintaining the environment in which tests are executed (adapted from [8]).

Test Execution

The test execution is the process of running the test procedure resulting from the test design and implementation process on the test environment established by the test environment set-up and maintenance process. The test execution process may need to be performed a number of times as all the available test procedures may not be executed in a single iteration (adapted from [8]).

Test Incident Reporting

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 8 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

The test incident reporting is the process of managing the test incidents. This process will be entered as a result of the identification of test failures, instances where something unusual or unexpected occurred during test execution, or when a retest passes (adapted from [8]).

1.2 SECURITY TESTING

In this section, we define a generic process for security testing based in the Technical Guide to Information Security Testing and Assessment from the National Institute of Standards and Technology (NIST) [14]. The NIST guide distinguishes between three typical phases of a security assessment and testing process. The phases are (Test-) Planning, (Test-) Execution and Post (Test-) Execution.

Planning

The Planning phase is used to gather all relevant information that is used during a security assessment. The main tasks of this phase are developing a security assessment policy, prioritizing and scheduling assessments, selecting and customizing technical testing and examination techniques, developing the assessment plan and addressing any legal considerations with respect to the security assessment. According to the NIST guide “a security assessment should be treated as any other project, with a project management plan to address goals and objectives, scope, requirements, team roles and responsibilities, limitations, success factors, assumptions, resources, timeline, and deliverables” [14].

Execution

Execution refers to the activity of executing tests in order to identify vulnerabilities, but also execute "scans" in order to obtain more information about the target of analysis (or system under test). That is, the NIST guide explicitly distinguishes between

- the analysis of the target (Target Identification and Analysis Techniques) by means of network discovery techniques, network port and service identification, vulnerability scanning techniques etc., and
- the validation of vulnerabilities (Target Vulnerability Validation Techniques) e.g. by means of password cracking, penetration testing, and social engineering.

Post-execution

The Post-Execution phase treats the findings from the previous phase. The NIST guide proposes a set of possible actions that range from simple reporting that is used to inform about the findings to developing mitigation actions and recommendations that help to mitigate the findings or prevent their exploitation. The activities in the post-execution phase are not exclusively part of the testing process but are also part of a risk assessment process that make use of the testing results.

With respect to the testing process described in the previous section,

- the Planning phase corresponds to the Test planning phase of the testing process;
- the Execution phase corresponds to the three activities Test Design & Specification, Test Environment Set-up and Maintenance, and Test Execution of the testing process;
- the Post-execution phase corresponds to the Test incident reporting activity of the testing process.

One of the main differences between the testing process described in the previous section and the NIST security testing process is that more emphasis is put on analysing the system under test by e.g. means of network discovery techniques. This can be seen even more clearly in the NIST process for penetration testing (which can be seen as a refinement of the above process) shown in Figure 3, where the test execution phase is split into two activities: *discovery* and *attack*. The testing process described in the previous section does not have any explicit activity that corresponds to the Target Identification or the discovery activity of the penetrating-testing process.

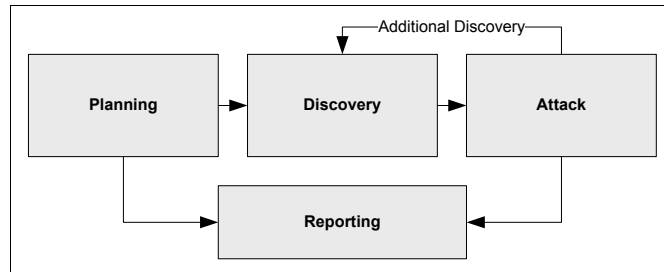


Figure 3 Four-stage penetration testing methodology

1.3 MODEL-BASED TESTING

In this section we describe the typical steps involved in a model-based testing process based on the ETSI document Methods for Testing & Specification (MTS); Model-Based Testing (MBT); Requirements for Modelling Notations [15].

As illustrated in Figure 4, model-based testing involves the following major activities: modelling for test generation, test selection criteria definition, test generation, test adaptation, and test execution.

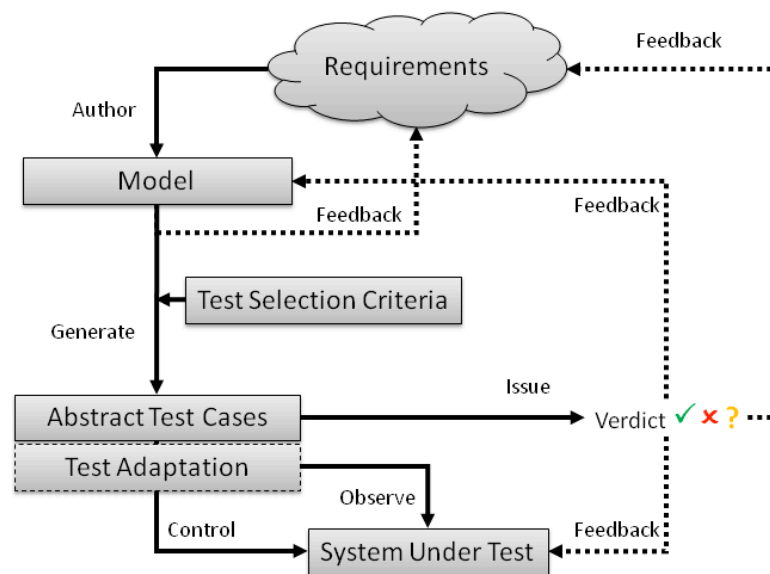



Figure 4 A generic model-based testing process

Modelling for Test Generation

The activity of defining the test model (i. e. computer-readable behavioural model that describes the intended external operational characteristics of a system, i.e. how the system being modelled interacts with its environment, in terms of the system interface) from which tests will be generated.

Test selection

The process or the result of choosing a subset of tests during test generation from a larger or infinite set of tests which can be derived from a model.

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 10 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

Test generation

The automatic derivation of abstract test cases in one or more different formats from a model based on user defined test selection criteria.

Test adaptation

The process of making the abstract test cases that are generated from the test models into concrete tests that can be executed.

Test execution

The process of execution the concrete test cases.

The main differences between the generic testing process (described in Section 1.1) and the model-based testing process described in this section, is that

- The activity test design has been split into modelling for test generation, test selection, and test generation
- The activity test implementation is referred to as test adaptation.

1.4 RISK ASSESSMENT

In this section, we describe a generic risk assessment process based on the ISO 31000 standard for risk management [7]. The overall risk management process shown in Figure 1 is taken from [7].

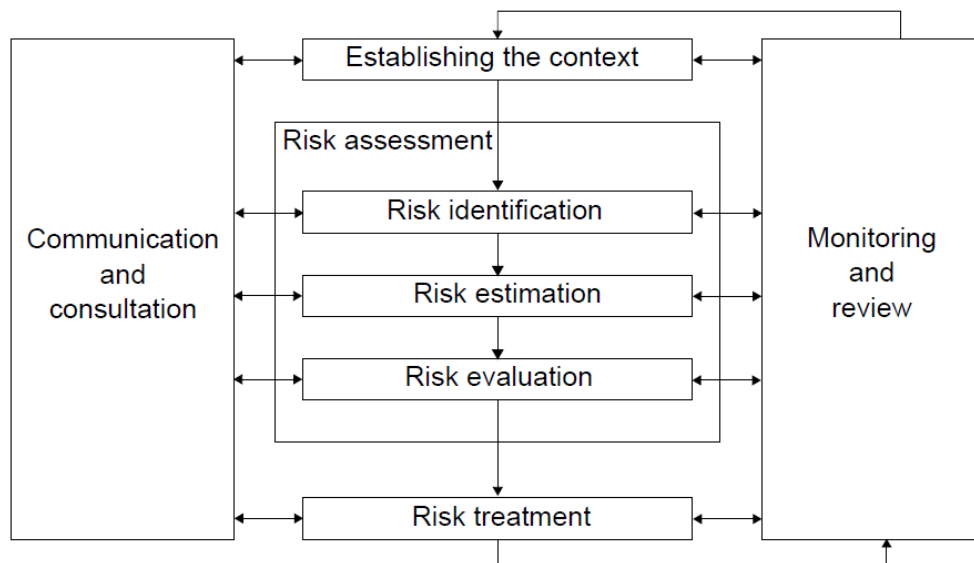



Figure 5 A generic risk management process

As stated in ISO 31000 [7], all activities of an organization may involve risk. Organizations usually manage risk by identifying it, estimating it and then evaluating it to see whether the risk should be modified by risk treatment in order to satisfy the risk evaluation criteria. Through this process, communication and consultation are carried out with stakeholders to monitor and review the risk, and controls that are modifying the risk are implemented to ensure that no further risk treatment is required. Risk management can be applied to an entire organization, at its many areas and levels, at any time, and to specific functions, projects and activities. Although the practice of risk management has been developed over time and within many sectors in order to meet diverse needs, the adoption of consistent processes within a comprehensive framework can help to ensure that risk is managed effectively, efficiently and coherently across an organization.

In the context of the DIAMONDS project, we focus on the five steps in the middle of Figure 5.

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 11 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

Establishing the Context

Establishing the context refers to the process of defining the external and internal parameters to be taken into account when managing risk, and setting the scope and risk criteria for the remaining process (adapted from [7]).

Risk Assessment

Risk assessment is the overall process of risk identification, risk estimation and risk evaluation (adapted from [7]). We sometimes also refer all five steps in the middle of Figure 5 as risk assessment.

Risk Identification

Risk identification is the process of finding, recognizing and describing risks. This involves identifying sources of risk, areas of impacts, events (including changes in circumstances), their causes and their potential consequences. Risk identification can involve historical data, theoretical analysis, informed and expert opinions, and stakeholder's needs [7].

Risk Estimation


Risk estimation is the process of comprehending the nature of risk and determining the level of risk. This involves developing an understanding of the risk. Risk estimation provides the basis for risk evaluation and decisions on whether risks need to be treated, and on the most appropriate risk treatment strategies and methods (adapted from [7]).

Risk Evaluation

Risk evaluation is the process of comparing the results of risk estimation with risk criteria to determine whether the risk and/or its magnitude is acceptable or tolerable. Risk evaluation assists in the decision about risk treatment (adapted from [7]).

Risk Treatment

Risk treatment is the process of modifying risk which can involve risk mitigation, risk elimination or risk prevention (adapted from [7]).

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 12 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

2. A GENERIC METHODOLOGY FOR MODEL-BASED SECURITY TESTING

In the following, we first (in Section 2.1) define a generic process for model-based security testing based on the discussion in Section 1, then (in section 2.2) we relate the activities of the process to the techniques developed in the DIAMONDS project.

2.1 A GENERIC PROCESS FOR MODEL-BASED SECURITY TESTING

In this section, we define a generic process for model-based security testing which combines the areas of testing, security testing, model-based testing, and risk assessment.

The steps of the process are shown in Figure 6 on the right hand side. The process is based on the overview given in Section 1 and in particular the generic testing process (described in Section 1.1). The main differences between the testing process and the generic process for model-based security testing are:

- The two activities test identification/discovery and test selection/prioritization have been added between the activities of test planning and test design & implementation. There are two reasons why these activities have been introduced into the process: (1) many processes for security testing have an explicit activity called "discovery" which is missing from the testing process. Since we are targeting security testing we therefore think that this activity should be part of the process. (2) Risk assessment results can be used in order to identify and prioritize tests before the test design takes place, and this is captured by the activities 2.a and 2.b in the generic process.
- The activity called test design & implementation of the testing process has been decomposed into four activities in the generic model-based security process (test specification/modelling, test generation, test selection/prioritization, and test adaptation/implementation). The reason for this is that the distinction between modelling / test generation / test selection is very important in order to highlight activities that are particular for *model-based* testing. Notice also that we have used the terms "test specification/modelling" instead of "test design" and "test adaptation/implementation" instead of "test implementation" to further highlight the focus on model-based testing.
- The activity 6.b test selection/prioritization has been added in parallel with the test execution activity. The reason for this is that test selection and prioritization is more important in security testing than functional testing since security testing typically generates a lot more test cases. In addition to this, techniques such as fuzzing can also be used on the execution level in order to mutate the test cases that are executed. Furthermore, it is also possible to use risk assessment results in order to select/prioritize test cases at the execution level.

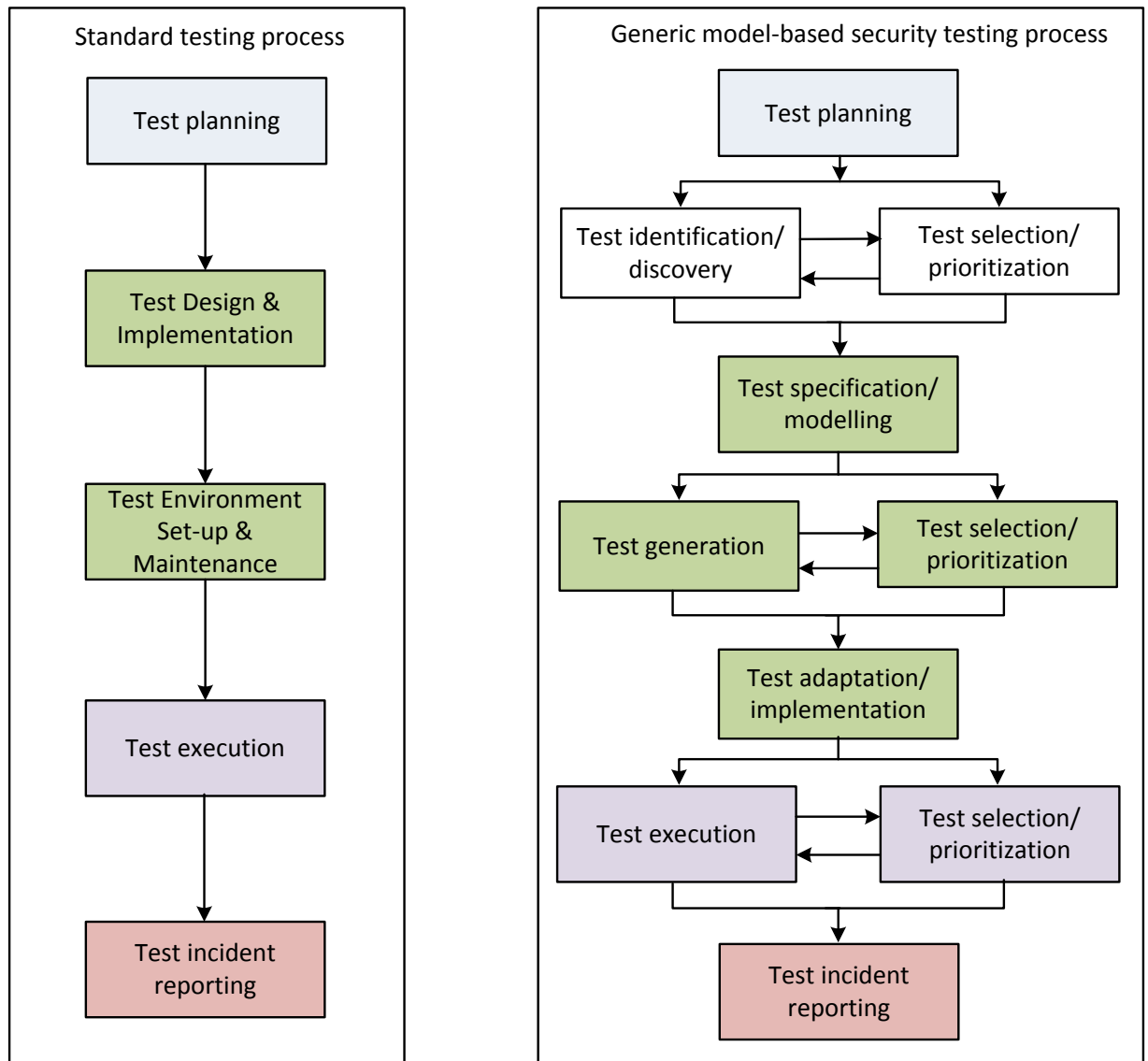


Figure 6 The generic process for model-based security testing and its relation to the testing process

In the following, we describe the steps of the process in more detail.

Step 1: Test planning

The test planning is the activity of developing the test plan. Depending on where in the project this process is implemented this may be a project test plan or a test plan for a specific phase, such as a system test plan, or a test plan for a specific type of testing, such as a performance test plan (adapted from [8]).


Step 2.a: Test identification/discovery

Test identification/discovery is the activity of identifying/discovering test scenarios or areas or vulnerabilities in the systems where the testing should be focused. The discovery activity may be performed by e.g. use of network discovering techniques, vulnerabilities scanners, or through risk assessment.

Step 2.b: Test selection/prioritization

The activity of prioritizing and selecting potential test scenarios that are identified in step 2.a.

Step 3: Test specification/modelling

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 14 of 39
		Version: 1.0
		Date : 16.05.2013
		Status : Final Confid : Public

The activity of defining the model for test generation (i. e. computer-readable behavioral model that describes the intended external operational characteristics of a system, i.e. how the system being modelled interacts with its environment, in terms of the system interface) from which tests will be generated.

Step 4.a: Test generation

The automatic derivation of abstract test cases in one or more different formats from a model based on user defined test selection criteria.

Step 4.b: Test selection/prioritization

The process or the result of choosing a subset of tests during test generation from a larger or infinite set of tests which can be derived from a model.

Step 5: Test adaptation/implementation

The process of making the abstract test cases that are generated from the test models into concrete tests that can be executed.

Step 6.a: Test execution

The test execution is the process of running the test procedure resulting from the test design and implementation process on the test environment established by the test environment set-up and maintenance process. The test execution process may need to be performed a number of times as all the available test procedures may not be executed in a single iteration (adapted from [8]).

Step 6.b: Test selection/prioritization

The activity of prioritizing and selecting tests to be executed. The selection criteria may e.g. be based on a risk assessment. The activity may also involve mutation/fuzzing of concrete executable test cases.

Step 7: Test incident reporting

The test incident reporting is the process of managing the test incidents. This process will be entered as a result of the identification of test failures, instances where something unusual or unexpected occurred during test execution, or when a retest passes (adapted from [8]). In test-based risk assessment, the incident reporting activity may involve an assessment of how the test results impact the risk picture.

2.2 RELATING DIAMONDS TECHNIQUES TO THE METHODOLOGY

In this section, we relate the DIAMOND techniques that are developed in work package 2 to the activities of the generic model-based security testing process.

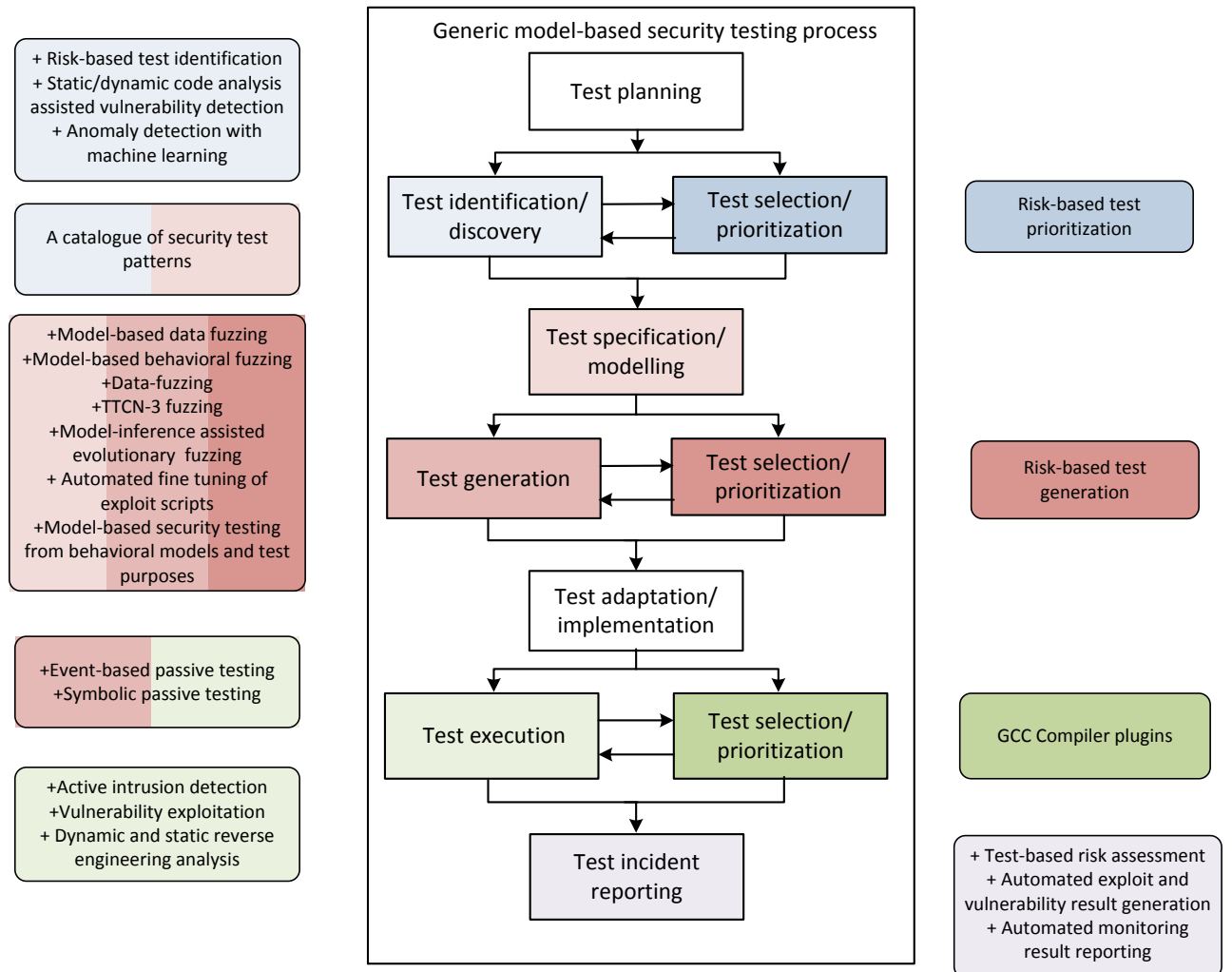



Figure 7 Relating DIAMONDS techniques to the generic process

As shown in Figure 7, the techniques developed in the DIAMONDS project support all the activities of the process except "Test planning" and "Test adaptation/implementation".

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 16 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

3. A SPECIFIC METHODOLOGY FOR MODEL-BASED SECURITY TESTING

3.1 OVERVIEW: COMBINED TMSR AND RMST

There are basically two different ways how model-based security testing and security risk analysis can be combined [23]. In Test-driven Model-based Security Risk Analysis (TMSR), security testing is carried out twice during the linear security risk analysis process, once to identify risks and once to evaluate them. In contrast, in Risk-driven Model-based Security Testing (RMST), risk analysis is twice used in the linear security testing process, once to identifying the security critical parts of the system under test and once to select the most important test cases (prioritization). The first approach tries to improve the security risk analysis with the help of security risk testing and the final output results are risk analysis artefacts. The second approach tries to improve the security testing with the help of security risk analysis and the final results are test result reports.

However, it makes sense to combine both approaches into a single process in order to get the most precise and complete picture. TMSR and RMST can benefit from one another in such a combined process. Indeed it might be helpful to switch multiple times between security testing and security risk analysis because after each round of testing and transferring the test results back into the risk picture, the risk analysis might be more precise and thus allow a better identification and prioritization of the next test cases. Such an iterative process is not linear, it is an incremental process that can be visualized as a cycle.

For the Risk-driven Model-based Security Testing (RMST) methodology described here, it is assumed that it is used in a combined process of TMSR and RMST as shown in Figure 8.

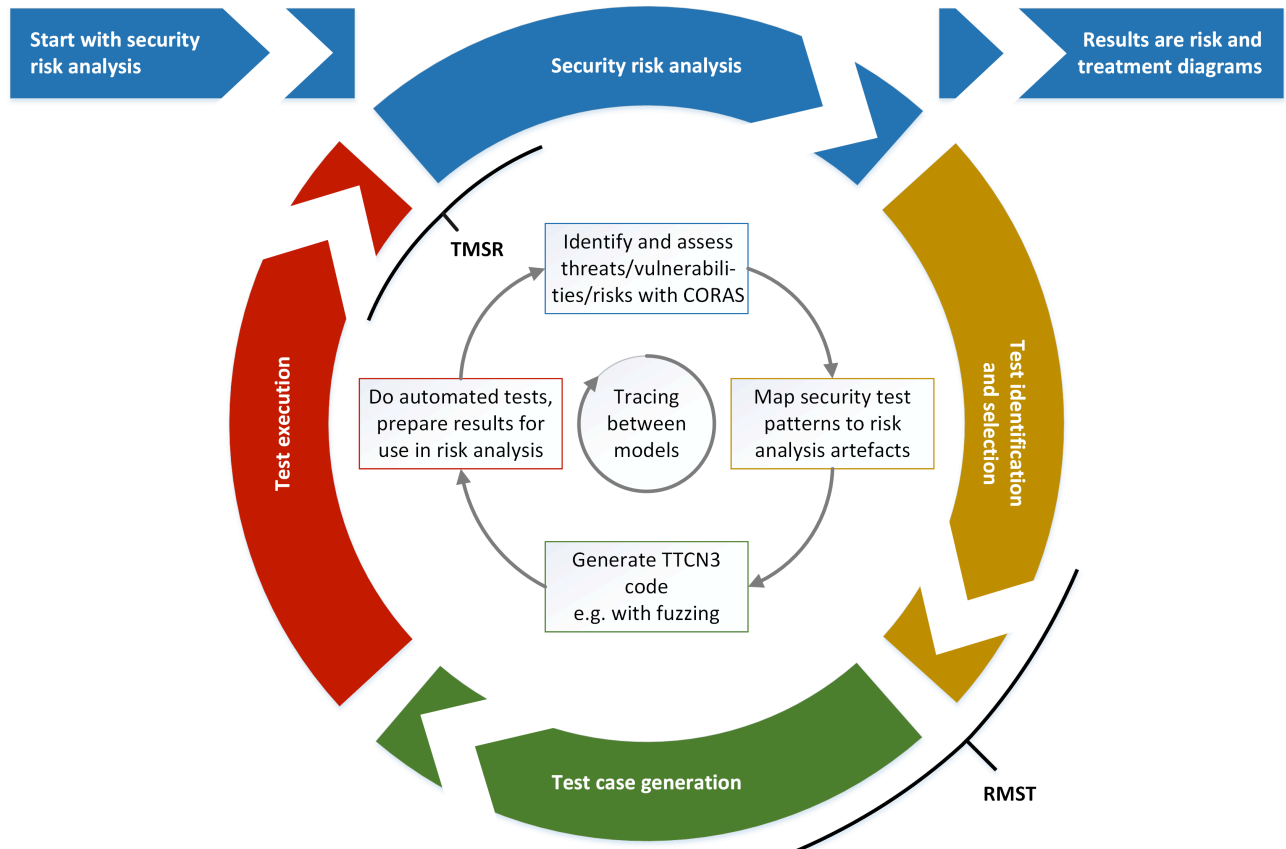


Figure 8 Model-based security testing process

Note that security risk analysis is seen as both the starting point and the end point for the combined TMSR and RMST process. The main reason for this design decision is that risk analysis might also include aspects that cannot be tested while all test results can be regarded as risk analysis results, too.


3.2 FROM RISK ANALYSIS ARTEFACTS TO TEST PATTERNS

Security risk analysis is conducted using the model-based CORAS method [24][9]. The CORAS method is performed till CORAS step 6, i.e. risk estimation with threat diagrams. The results of this analysis are expressed with threat diagrams containing likelihood and consequence values. This initial analysis is based on literature, vulnerability databases and the system model. Its results are highly dependent on the experience and the skills of the risk analysis team. Important aspects might have been missed completely and the just guessed likelihood values are eventually very uncertain.

3.2.1 Selecting elements to test

Though the initial analysis is probably somehow imprecise and not very reliable, it is a good starting point for the first round of the security testing process. Instead of just guessing how likely the potential security risks are, testing provides reliable results that can be interpreted as indicators for these likelihood values. Security testing can be used to measure to what extent the so far identified vulnerabilities can be exploited as described in the threat scenarios to trigger the identified unwanted incidents.

While the threat diagram immediately can be interpreted as a guide telling the analysts what should be tested, it is not obvious which tests are the most critical and which tests will probably not have a significant impact on the overall risk picture. Since security testing can be expensive and since often both time and resources available for testing are rather limited, it would be most helpful to identify the most relevant test cases and to test these in the first place.

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 18 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

Within the combined TMSR and RMST process, the threat diagram is used to identify nothing but the most critical threat scenario that has not yet been tested. There are two different methods to do so.

The first algorithm tries to evaluate the risk for each threat scenario using appropriate risk functions. Risk functions have two input parameters: a likelihood value and a consequence value. In CORAS, consequence values are typically only assigned to relations leading from unwanted incidents to assets. A threat scenario might have multiple unwanted incidents and each incident might have multiple relations to assets having different consequence values. For each path in the threat diagram leading from a threat scenario to an asset and having some consequence value, there might be a different likelihood value. Applying an appropriate risk function for each path, a risk value for that path can be calculated. If all these risk values are on the same scale and if that scale allows to add risk values in a reasonable way, then it becomes possible to calculate a single accumulated risk value for each entire threat scenario by summing up all the risk values for the paths to assets. In order to make sure that the accumulated risk values for different threat scenarios are comparable, all risk functions must use the same global scale of risk values and there must be an order within the risk values.

The second algorithm uses multiple simulations to identify the most critical threat scenario that has not yet been tested. For each simulation, it changes the likelihood for some not yet tested threat scenario to zero to analyse the impact of that threat scenario being absent for the entire risk picture by applying the regular CORAS risk evaluation (CORAS step 7) for each of these simulation threat diagrams. Finally the results for the different simulations are compared. Therefore, all risk functions must use the same global scale of risk values, it must be possible to accumulate the risk values in a sound way and there must be an order within the risk values. The simulation showing the least critical overall risk picture is exactly the simulation that was created by setting the likelihood of the most critical likelihood to zero.

In the combined TMSR and RMST process, only a single threat scenario that should be tested next will be selected. Then that threat scenario will be tested and the result will be used to improve the threat diagram before trying to identify the next most critical threat scenario which was not yet tested.

3.2.2 Map security test patterns to threat scenarios

Knowing what should be tested next is fine. However, it can be challenging to create effective test cases and create appropriate metrics that allow sound conclusions for the likelihood values in the threat diagrams. Instead of reinventing the wheel each and every time, it makes sense to create and to use a catalogue of test patterns [25].

Security test pattern do typically consist at least of a name, a context, a problem and a solution description. For the combined TMSR and RMST process, the threat scenario is a direct counterpart to the problem description of a test pattern. Hence it is easy to identify a fitting test pattern for the most critical threat scenario that needs to be tested if such a pattern already exists in some database.

The solution description of a test pattern will typically contain some generic input parameters and some output results. The input parameters should be mapped to the vulnerabilities that have relations leading to the most critical threat scenario in the threat diagram. Vulnerabilities will be used as the input ports to pass test values to the system under test. The output results can be mapped to the unwanted incidents that might be triggered if the threat scenario comes true. Figure 9 shows an example mapping.

Eventually the same test pattern will have to be instantiated multiple times for a single threat scenario because there are different vulnerabilities and unwanted incidents it can be mapped to.

3.3 FROM TEST PATTERNS TO TEST IMPLEMENTATION AND EXECUTION

Implementation errors are not only a problem of the system under test. Test cases can also be implemented so that they are not effective – or that they produce wrong results, which would be much worse. A test pattern contains at least a brief description how the test should be implemented and thereby eventually helps to prevent some potential implementation errors. Probably a test pattern contains even some generic template code that can be instantiated into executable code.

For effective security testing, often lots of different test cases have to be generated that are close to the limits of valid input sequences. Typically, in such a case not all test cases are created manually. Instead, model based fuzzing can be used to generate appropriate random test sequences.

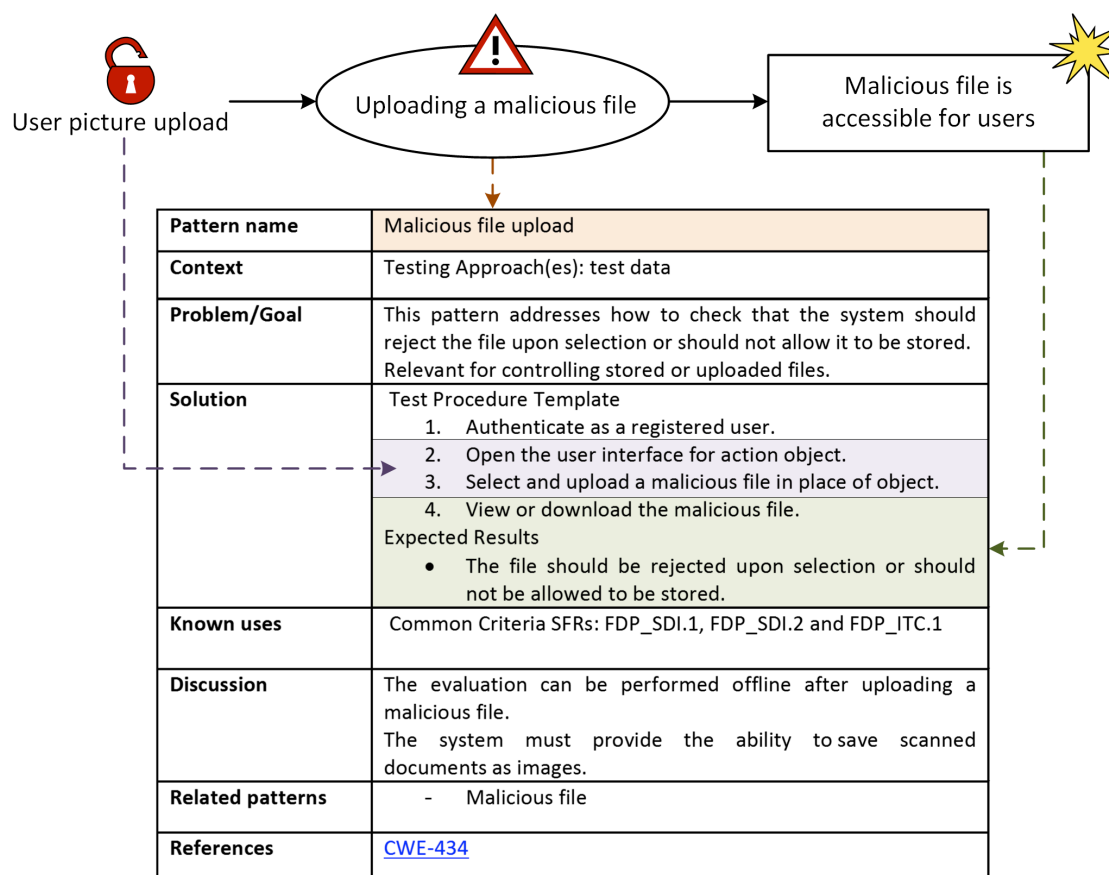


Figure 9: Mapping threat diagram artefacts to test pattern

The specific model-based security testing methodology described here suggests TTCN-3 to specify the test cases in a flexible and implementation independent way. Not only the test cases, but also the test patterns can be created in a flexible way using TTCN-3 notation [22] and it makes sense to use the same notation for both. There is already tool support available for automatic test-case generation producing TTCN-3 code, e.g. for model-based fuzz testing the library developed within the DIAMONDS project, see D5 WP3.

To apply the test cases and to aggregate the results, any compatible test execution environment may be used. Within the DIAMONDS project, for example the commercial TWorkbench from Testing Technologies and CertifyIt from Smartesting have been used successfully as the test development and execution environment for the Risk-driven Model-based Security Testing methodology described here. A list of additional test tools with TTCN-3 support can be found at <http://www.ttcn-3.org>.

3.4 COMPLETE THE ITERATION WITH TMSR

The next step is the integration of the test results into the risk picture. Therefore, the CORAS process is started again at CORAS step 5 with the test results as new additional input information. These test results might bring vulnerabilities, threat scenarios and unwanted incidents to attention that were not recognized before. Additionally, the estimation of likelihoods (i.e. CORAS step 6) might become more precise taking the test results into consideration. This step is the actual Test-driven Model-based Security Risk Analysis (TMSR) and a specific methodology for TMSR will be discussed in section 4.

The risk picture can be iteratively improved by starting again with step 3.2.1 using the threat diagram updated with the latest test results as input.

4. A SPECIFIC METHODOLOGY FOR TEST-BASED RISK ASSESSMENT

In this section, we present an instance of the generic process for model-based security testing which is based on the CORAS risk assessment process [9]. The process can be seen as an extension of the CORAS process (which does not specifically take testing into account) into a process that takes testing explicitly into account.

Many of the activities that we describe in the following are based on the CORAS methodology, and the reader is referred to [9] for more details on these. However, to keep this document self-contained, we briefly explain and give examples of those CORAS activities that are needed to understand the process.

4.1 OVERVIEW OF PROCESS AND INSTANTIATION

In the following, we first (in Section 4.1.1) give an overview of the main steps of the test-based risk assessment process, then we (in Section 4.1.2) describe how the process relates to the generic process for model-based security testing.

4.1.1 Overview of process

The main steps of the process for test-based risk assessment are shown in Figure 7. In the following, we briefly describe each of the steps.

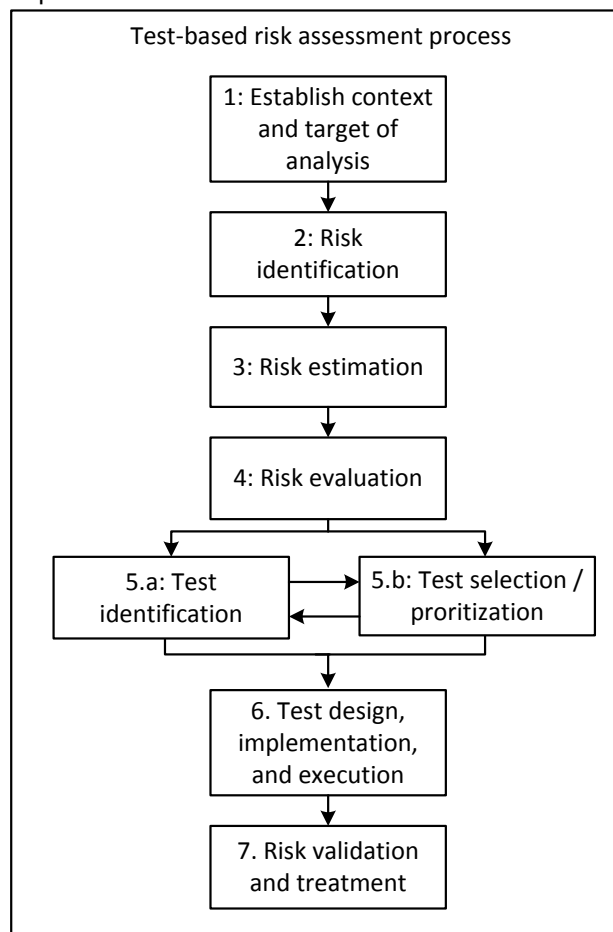



Figure 10 A specific test-based risk assessment process

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 21 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

1: Establish context and target of analysis.

This activity is based on Step 1 – Step 4 of the CORAS risk assessment methodology. The objective of the activity is to define (1) the target of analysis, (2) the assumptions, focus and scope of the analysis, (3) assets and stakeholders, (4) consequence and likelihood scales, and (5) risk evaluation criteria in the form of a risk matrix.

2. Risk identification

This activity corresponds to Step 5 of the CORAS methodology. The objective is to identify unwanted incidents, threats, threat scenarios and vulnerabilities w.r.t. to the identified assets, and to document these using CORAS threat diagrams.

3. Risk estimation

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

4: Risk evaluation

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

5.a : Test identification

The objective of this activity is to identify potential test scenarios that are described by the risk model specified in the previous activities.

5.b : Test selection/prioritization

The objective of this activity is to prioritize the potential test scenarios that are identified, and then select those test scenarios that will be implemented into executable test cases (in activity 6).

6.: Test design, implementation, and execution

The objective of this activity is to design, implement executable test cases based on the test scenarios that have been selected, and to execute the test cases and document the test results.

7.: Risk valuation and treatment

The objective of this activity is to validate the risk model based on the test results obtained in activity 6, and to suggest treatments for the most severe risks. The latter corresponds to Step 8 of the CORAS methodology.

4.1.2 Instantiation

In this section, we describe how the specific process for test-based risk assessment instantiates the generic model-based security testing process.

The relation between the two processes is illustrated in Figure 11, where the relations between the activities in the two processes are illustrated by the dotted lines. The main differences between the two processes are:

- In the specific process, test planning is referred to as "Establish context and target of analysis" which is more in line with the terms that are used in the risk assessment community.
- Activity 2.a. has been decomposed into 4 activities in the specific process (activities 2-4 and 5.a) to make the risk assessment steps performed prior to the test identification explicit.
- All activities involving test specification/modelling to test execution (activities 3 – 6.b) in the generic process are in the specific process merged into one activity called Test design, implementation, and execution. This is because the main emphasis of the specific process is on risk assessment, not testing.

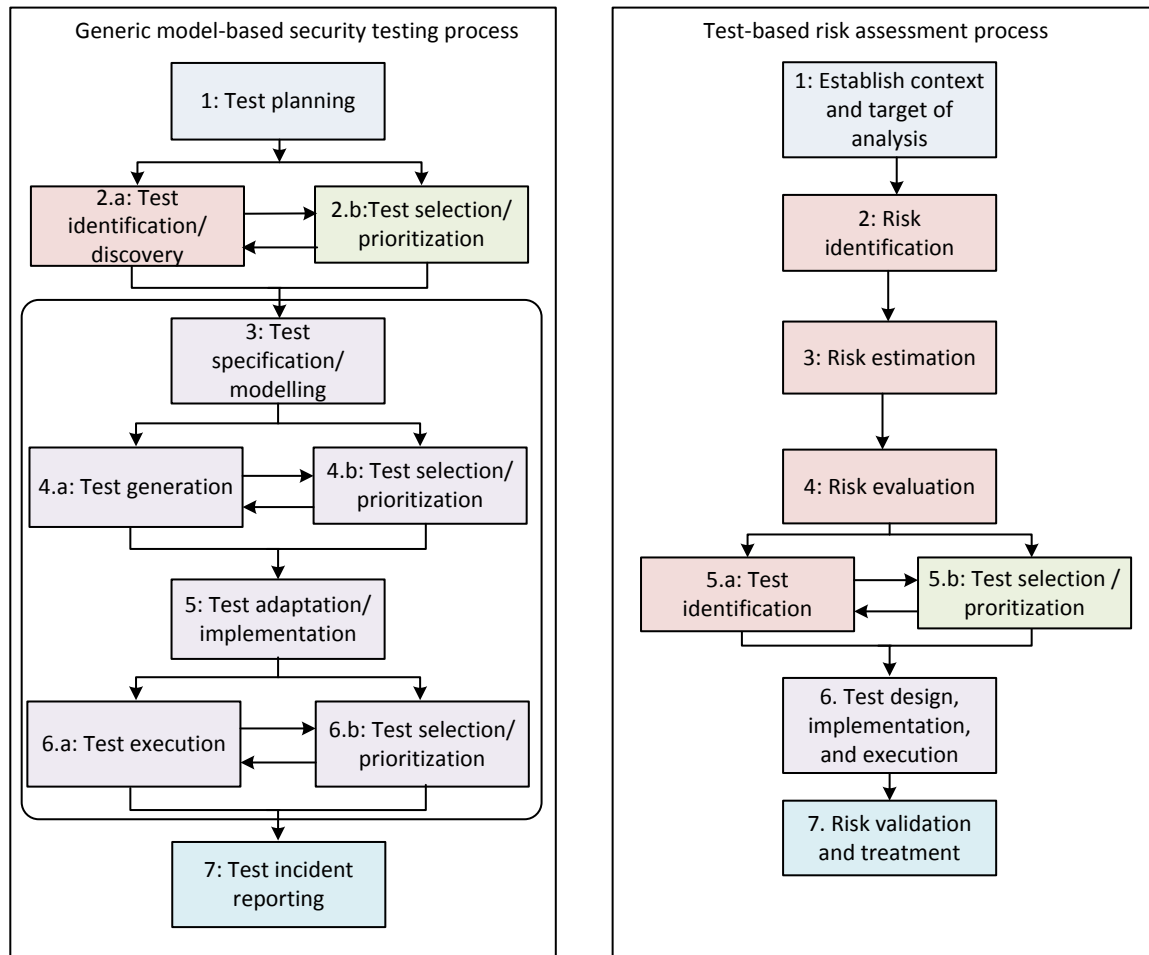


Figure 11 Relation the specific test-based risk assessment process to the generic process

4.2 DESCRIPTION OF THE PROCESS

In this section, we describe each of the activities of the process for test-based risk assessment in more detail.

4.2.1 1: Establish context and target of analysis.

This activity is based on Step 1 – Step 4 of the CORAS risk assessment methodology. The output of the activity 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, and
- 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 speak of, then there is no need to conduct a risk assessment.

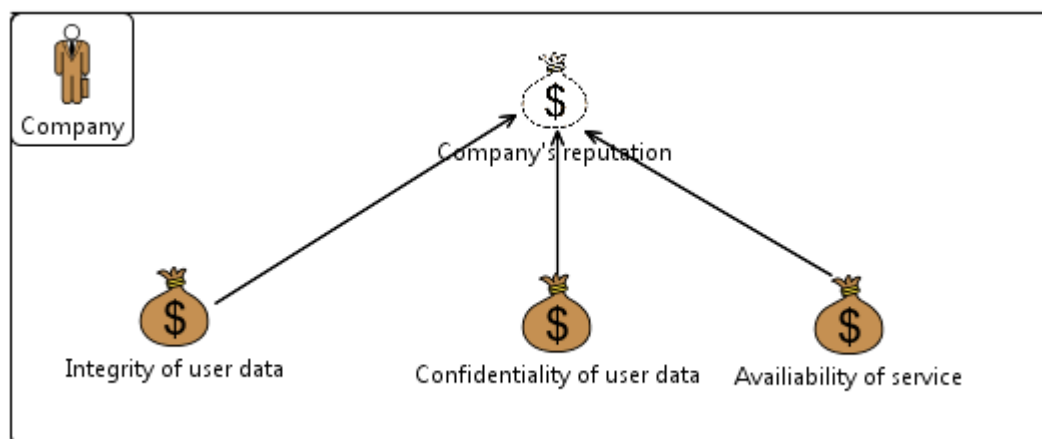


Figure 12 Example of a CORAS asset diagram

An example of a CORAS asset diagram is illustrated in Figure 12. The party (Company) which assigns values to the assets is specified in the top left corner of the diagram. In the diagram proper, three assets 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.

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	[20,49] : 10y
Possible	Once a year	[6,19] : 10y
Unlikely	Less than once per year	[2,5] : 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>

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 24 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

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 using risk matrixes. 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 analysis to define what is mean 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					

4.2.2 2: Risk identification

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

- A set of CORAS threat diagrams.

Risks should be identified in work shops where risks and their causes are systematically identified by work shop 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 whose nodes are of one the following kinds

- **Threat:** A potential cause of an unwanted incident (illustrated by a man 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 13.

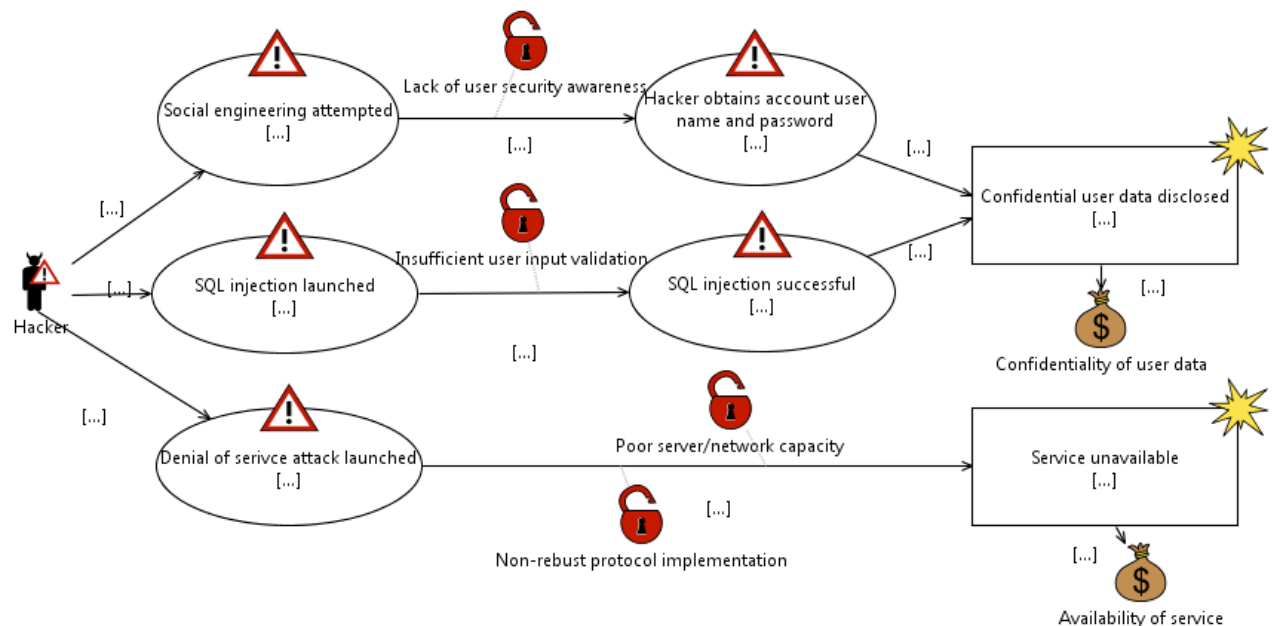


Figure 13 Example of a CORAS threat diagram

4.2.3 3: Risk estimation

This activity corresponds to step 6 of the CORAS methodology. The objective is to identify (1) likelihood values for threat scenarios and unwanted incidents, (2) conditional likelihoods between risk elements, (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 work shops where the work shop participants estimate likelihood values based on expert judgment and/or historical data.

An example of a CORAS threat diagram with likelihood and consequence values is given in Figure 14. 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 activity 1.

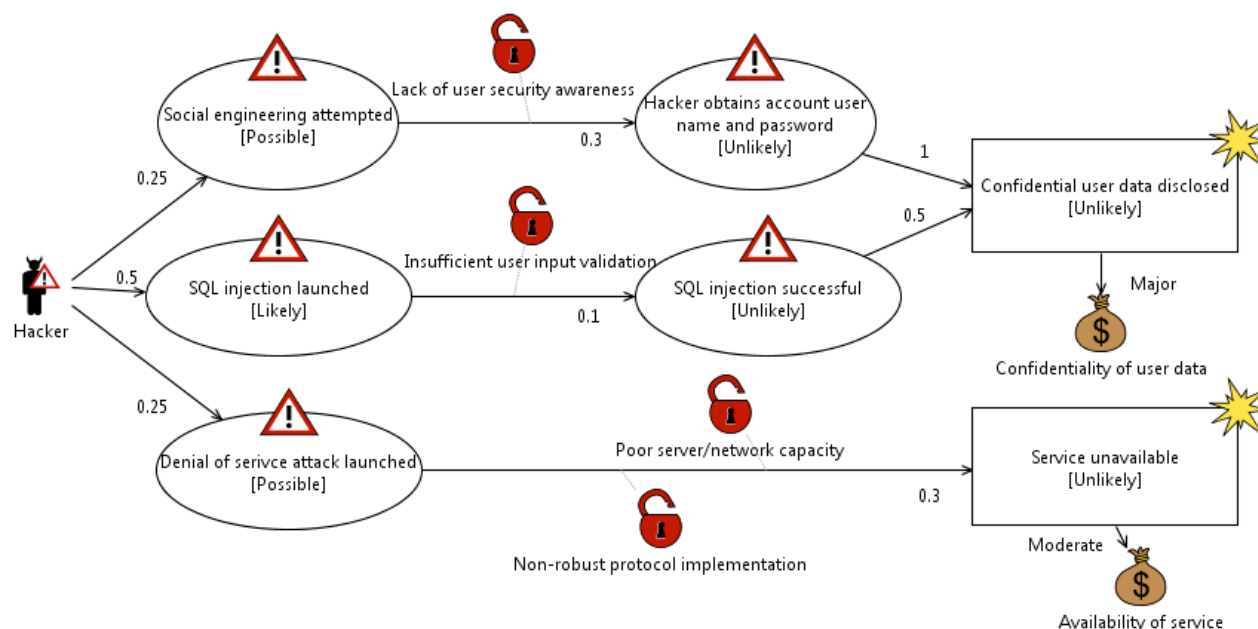


Figure 14 Example of a CORAS threat diagram with likelihood values

4.2.4 4: Risk evaluation

This activity corresponds to Step 7 of the CORAS methodology. The objective is to identify and prioritize/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 activity 1).

Table 5 shows an example of a risk evaluation matrix where the two risks shown in Figure 14 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					
	Minor					
	Moderate		Service unavailable			
	Major		Confidential user data disclosed			
	Catastrophic					

4.2.5 5.a: Test identification

The objective of the test identification activity is to identify potential test scenarios based on the CORAS threat diagrams that have been specified up to this point. The output of the activity is

- A table describing all potential test scenarios of the CORAS risk diagrams.

There are two threat diagram elements in particular that are useful for identifying possible test scenarios:

- *Threat scenario*, because they say something about *how* to test. Threat scenarios typically describe some sort of misuse-case or attack, and this can be used as a basis for security testing.
- *Vulnerability*, because they say something about what to look for when testing. Indeed, test results will usually either confirm the presence of a potential vulnerability, or return inconclusive in the case that the vulnerability was not found.

One possibility is to let each test scenario correspond to a threat scenario or a vulnerability. However, the problem with this is that the relationship between vulnerabilities and threat scenarios that are in the diagram is then lost. We therefore take relations in the CORAS threat diagrams as the basis for test identification. In particular, we take the approach that each initiates and leads-to relation in the diagram corresponds to a potential test scenario (since both kinds of relations may involve threat scenarios and vulnerabilities).

The CORAS approach has a procedure for schematically translating CORAS diagrams into English. When describing the potential test scenarios of a CORAS diagram, we use this to translate every initiates and leads-to relation in the diagram into English.

An example is given in Table 6, showing a list of potential test scenarios generated on the basis of the threat diagram in Figure 14. Here we see the English translation of every initiates and lead to relation described in the CORAS threat diagram shown in Figure 14.

Table 6 Example of identified test scenarios

Id	Test scenario
TS1	<i>Hacker initiates Social engineering attempted with likelihood 0.25.</i>
TS2	<i>Hacker initiates SQL injection launched with likelihood 0.5.</i>
TS3	<i>Hacker initiates Denial of service attack launched with likelihood 0.25.</i>
TS4	<i>Social engineering attempted leads to Hacker obtains account user name and password with conditional likelihood 0.3, due to Lack of user security awareness.</i>
TS5	<i>SQL injection launched leads to SQL injection successful with conditional likelihood 0.1, due to Insufficient user input validation.</i>
TS6	<i>Denial of service attack launched leads Service unavailable with conditional likelihood 0.3, due to Poor server/network capacity and Non-robust protocol implementation.</i>
TS7	<i>Hacker obtains account user name and password leads to Confidential user data disclosed with conditional likelihood 1.</i>
TS8	<i>SQL injection successful leads to Confidential user data disclosed with conditional likelihood 0.5.</i>

4.2.6 5.b: Test selection/prioritization

The purpose of the test selection/prioritization activity is to prioritize the identified potential test scenarios, and based on this, select the test scenarios that will developed further into concrete test cases. The output of the activity is

- A table with a prioritized list of test scenarios with an indication selected test scenarios.

The details of how to calculate the priority of given test scenario is described in the final deliverable of WP2. In the following, we give a brief overview.

The priority of a given test scenario is calculated based on three values:

- **S (Severity)**: An estimate of the impact that the test scenario has on the risks. As explained in the WP2 deliverable, this value is calculated by comparing the risks of two risk models: one model where the conditional likelihood of the relation corresponding to the test scenario is set to the minimum likelihood value (specifying that the relation will never lead up to something), and one model where the conditional likelihood of the relation corresponding to the test scenario is set to the maximum likelihood value (specifying that the relation will always lead up to something). The severity S is

then the difference between the two risk models. A high severity value suggests that the test scenario has a high impact on the risk, and that it therefore should be prioritized for testing.

- **T (Testability):** An estimate of how testable a test 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 we are about the correctness of the conditional likelihood value of the relation corresponding to a test 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.

Both the uncertainty and the testability values must be supplied by the user. The severity value on the other hand, can be calculated based on the likelihood and the consequence values that are already in the risk model as described in the final deliverable of WP2.

In Figure 15, we show an example of a CORAS threat diagram where all initiation and leads-to relations are annotated by labels of the form T:X, U:Y, specifying that the corresponding test scenario has a testability value X and an uncertainty value Y. In the figure, both testability and uncertainty values are assumed to be between 0 and 4.

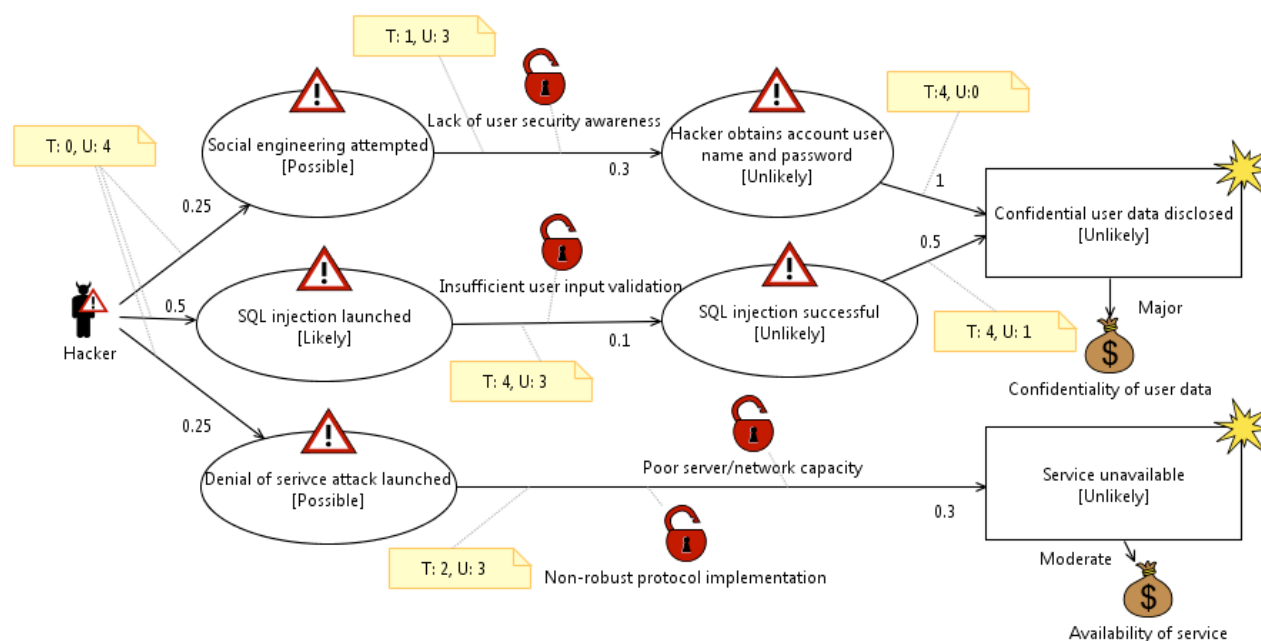


Figure 15 Example of a CORAS threat diagram with annotations for testability and uncertainty

Having annotated the risk model with testability and uncertainty values. A table containing a prioritized list of test scenarios can be generated. An example of such a table is shown in Table 7. Here all test scenarios correspond to initiates and leads-to relation in Figure 15, and values for testability (T) and uncertainty (U) have been taken from the annotation in the diagram. The severity value (S) can be calculated automatically using the procedure described in the deliverable of work package 2. In the example shown in Table 7, dummy values have been inserted for severity (i.e. we have not calculated them).

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 product of S, T, and U. However, other functions are possible.

Based on the list of prioritized test scenarios, the user must select the test 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 test scenarios that have a higher priority than *PT*.

In Table 7, the two first test scenarios with the highest priority values are shown in boldface, indicated that these two test scenarios are selected for further testing, and the others not.

Table 7 Example of a prioritized list of test scenarios

Id	Test scenario	S	T	U	Priority
TS5	SQL injection launched leads to SQL injection successful with conditional likelihood 0.1, due to Insufficient user input validation.	3	4	3	36
TS6	Denial of service attack launched leads Service unavailable with conditional likelihood 0.3, due to Poor server/network capacity and Non-robust protocol implementation.	3.2	2	3	19.2
TS4	Social engineering attempted leads to Hacker obtains account user name and password with conditional likelihood 0.3, due to Lack of user security awareness.	1.5	1	3	4.5
TS1	Hacker initiates Social engineering attempted with likelihood 0.25.	2.5	0	4	0
TS2	Hacker initiates SQL injection launched with likelihood 0.5.	2.5	0	4	0
TS3	Hacker initiates Denial of service attack launched with likelihood 0.25.	2.5	0	4	0
TS7	Hacker obtains account user name and password leads to Confidential user data disclosed with conditional likelihood 1.	1	4	0	0
TS8	SQL injection successful leads to Confidential user data disclosed with conditional likelihood 0.5.	2	4	0	0

4.2.7 6: Test design, implementation, and execution

The objective of activity 6 is to design, implement and execute test cases for each of the test scenarios and each vulnerability in the test scenarios that were selected in the activity 5:

- The output of the activity is a table which describes the test results for each selected test scenario and vulnerability.

Our process does not stipulate the manner in which test are designed, implemented, and executed. However, the test results should be documented in a manner that makes it convenient to assess their impact on the risk model. To this end, we suggest that the test results be documented in a table consisting of the following columns:

- Test scenario ID:** The of the test scenario that was tested.
- Vulnerability:** A vulnerability related to the test scenario that was tested.
- Likelihood:** A value indicating the likelihood that the vulnerability that was tested is actually present in the system under test.
- Exploitability:** A value indicating how hard it is to exploit the vulnerability, or more precisely how likely it is that an attempt to exploit the vulnerability will result in the outcome of the test scenario, where outcome refers to the target threat scenario or unwanted incident of the relation from which the test scenario was derived.

An example of a test result report is given in Table 8. Here we see the two test scenarios that were selected in the example of activity 5 and their associated vulnerabilities. In the cases that the test results confirm the presence of vulnerabilities in the system, then the likelihood value should be 1 (indicating that it is 100% certain that the vulnerability exists). For instance, if the testing found vulnerability "insufficient user input validation", then its likelihood value should be set to 1 as shown in Table 8. If a tested vulnerability is not found by the tests, then this does not necessarily mean that the vulnerability does not exist because it is possible the tests failed to uncover it. In this case, the user must estimate the likelihood value based on experience and the test results, for instance by looking at the number of tests that were executed or some kind of test coverage measure.

Table 8 Example of test result report

Test scenario ID	Vulnerability	Likelihood	Exploitability
TS5	Insufficient user input validation	1	0.8
TS6	Poor server/network capacity	0.4	0.7
TS6	Non-robust protocol implementation	0.2	0.6

4.2.8 7: Risk validation and treatment

The objectives of activity 7 are to (1) validate the risk model based on the test results 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 validation and updating the CORAS model. Specifically, for each test scenario, the user must decide whether the likelihood and exploitability values of the vulnerabilities of the test scenario are consistent with the conditional likelihood value of the relation in the risk model from which the test scenario was derived. One way of doing this is to let the conditional likelihood value of a relation be equal to the sum of the likelihood value multiplied by the exploitability value for each vulnerability of the test scenario. If we follow this procedure based on the test report of Table 8, we get

- Conditional likelihood of TS5 = $1 * 0.8 = 0.8$
- Conditional likelihood of TS6 = $(0.4 * 0.7) + (0.2 * 0.6) = 0.4$

The CORAS risk model should be updated based on the new conditional likelihood values that were derived based on the test report. First, all the conditional likelihood values must be updated, and then the likelihood values of the threat scenario and unwanted incidents that depend on these conditional likelihood values must be recalculated using on the CORAS rules for likelihood calculation.

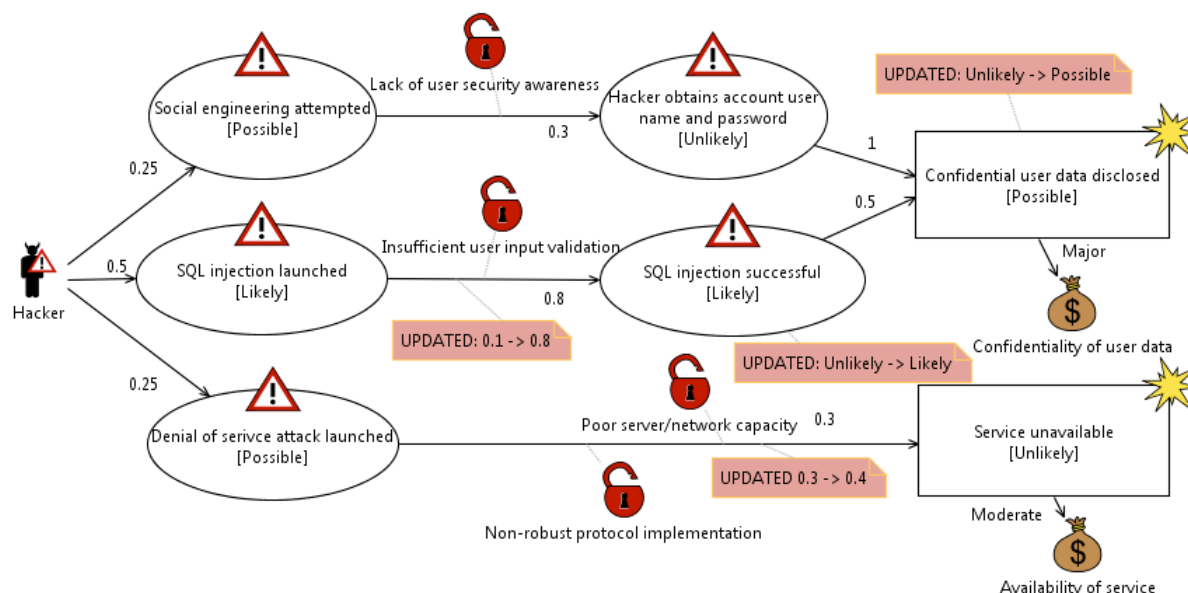


Figure 16 Example of an updated risk model based on test results

Figure 16 shows an example of an updated risk model. Here the risk model shown in Figure 15 have been updated based on the test results of Table 8. Specifically, the conditional likelihood value of the relation corresponding to test scenario TS5 has been changed to 0.8 (from 0.1), and the conditional likelihood value of the relation corresponding to test scenario TS6 has been changed to 0.4 (from 0.3). In addition, these changes have affected the likelihood values of some of the threat scenarios and unwanted incidents that are targeted by the affected relations. In particular, the likelihood value of threat scenario "SQL injection successful" has changed from "Unlikely" to "Likely", and the likelihood value of the unwanted incident "Confidential user data disclosed" has changed from "Unlikely" to "Possible".

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 16, then the corresponding risk is categorized as red when plotting in into the risk evaluation table (see Table 9). Comparing the previous risk evaluation matrix (Table 8) 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 9 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.

An example of a CORAS treatment diagram is shown in Figure 17. Here the treatment "Implement input validation mechanism" has been associated with the vulnerability "Insufficient user input validation".

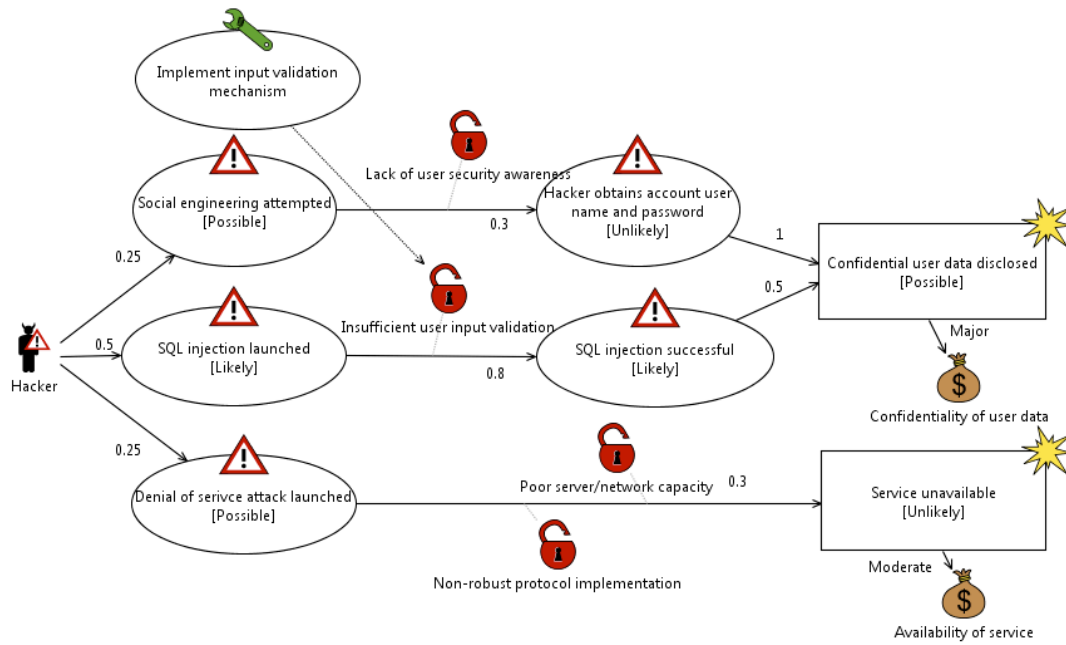



Figure 17 Example of a CORAS treatment diagram

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 33 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

5. A CATALOGUE OF SECURITY TEST PATTERNS

Patterns are an established approach for facilitating the reuse of known solutions, guidelines or best practices to recurring problems in various domains (software engineering, design, etc.). There have been some efforts for applying the same approach to testing and test automation [16][17]. Software security is another domain in which patterns have been gaining more popularity recently.

Although security test patterns are a relatively new research field, several definitions have been provided by the community. For instance, the SecurityTestPatterns group associated to MITRE's Common Weaknesses Enumeration (CWE) define them as: „*A software security test pattern is a recurring security problem, and the description of the a test case that reveals that security problem, that is described such that the test case can be instantiated a million times over, without ever doing it the same way twice.*“

While several definitions are now given, the notion of "design patterns" could be close to our used definition. A design pattern is a description of a recurring problem and a well-defined description of the core solution to the problem that is described such that the pattern can be used many times but never in exactly the same way [18].


Our notion is visible in the following question: "what is the pattern to test the security properties of an SUT?" We finally embrace both of the above definitions to define patterns that are, on the one hand, generic enough to adapt to various testing strategies and, on the other hand, include patterns for concrete security test cases.

We give in the following subsections an overview of the security test patterns defined by this work package and detailed in the deliverable D3.WP4.T1 as well as some of their uses by the DIAMONDS partners in the WP1 use cases.


5.1 OVERVIEW ON THE DIAMONDS SECURITY TEST PATTERNS

We present in this section a summary of the security test patterns that have been identified in DIAMONDS. We summarize the 17 test patterns by the following table by indicating the context used, specifically the testing approaches, the problems addressed by the patterns as well as some of the WP1 case studies in which some patterns have been used.

Pattern Name	Context	Problem	Case Studies
Attacking Authentication Mechanism	Testing Approach(es): detection, test data	This pattern addresses how to check that the system handles high number of authentication attempts with incorrect passwords. Relevant for authenticating	
Testing the safe storage of authentication credentials	Testing Approach(es): detection	This pattern addresses how to check that the system store in a safe way the user authentication information. Relevant for user authentication management	
Verify audited event's presence	Test Pattern Kind: Behavioral Testing Approach(es): Detection	This pattern addresses how to check that a system logs a particular type of security-relevant event for auditing purpose	
Verify default-authentication	Test pattern kind: Behavioral Testing Approach(es):	This pattern addresses how to check default authentication mechanisms.	

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 34 of 39
		Version: 1.0
		Date : 16.05.2013
		Status : Final Confid : Public

credentials to be disabled on production system	Prevention	Relevant for systems based on open-source software.	
Verify audited events' content	Test Pattern Kind: Behavioral Testing approach(es): Detection	This pattern addresses how to check that a system logs a particular type of security-relevant event for auditing purpose	
Verify presence/efficiency of prevention mechanism against brute force authentication attempts	Test pattern kind: Behavioral Testing Approach(es): Prevention, Detection	This pattern addresses how to check password brute-forcing attack on computing systems providing a password-based authentication scheme.	
Verify presence/efficiency of encryption of communication channel between authenticating parties	Test pattern kind: Behavioral Testing Approach(es): Prevention	This pattern addresses how to check Man-in-the-middle attacks. Relevant to protect the data exchange between authenticating parties from eavesdropping attempts.	
Usage of Unusual Behavior Sequences	Test pattern kind: Behavioral Testing Approach(es): Prevention	This pattern addresses some attacks (e.g. Authentication bypass) that may be possible by subjecting the system to a behavior sequence that is different from what would be normally expected. In certain cases, the system may be so confused by the unusual sequence of events that it would crash.	DCo (IT –Symbolic passive testing) Giesecke & Devrient
Detection of Vulnerability to Injection Attacks	Test pattern kind: Data Testing Approach(es): Prevention	This pattern addresses the test of information systems resilience to injection attacks to increase their security confidence level.	
Detection of vulnerability of data structure attacks	Test pattern kind: Data Testing Approach(es): Prevention	This pattern addresses how to detect vulnerabilities to data structure attacks. Relevant to avoid attacker manipulating and exploiting characteristics of system data structures.	Giesecke & Devrient
Session Management Attack	Testing Approach(es): behavioral and test data	This pattern addresses how to check that the system returns an authorization error when the session information is faked or forged, and that no sensitive information is returned after requests. Relevant for managing/controlling access the system.	
Redirect header manipulation	Testing Approach(es): design	This pattern addresses how to check that the system handles correctly the users redirection after authentication. Relevant for URL parameters rejection.	
Malicious file upload	Testing Approach(es): test data	This pattern addresses how to check that the system should reject the file upon selection or should not allow it to be stored.	

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 35 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

		Relevant for controlling stored or uploaded files.	
Search for documented passwords	Testing Approach(es): detection, test data	This pattern addresses how to check that the system should not list any default passwords or usernames that are hard-coded into the product.	
Impersonating trusted external resources	Testing Approach(es): design, data	This pattern addresses how to check that the system refuses (or behave as such) to connect an impersonated server. Indeed, by DNS spoofing or DNS entry modifications, the authentic external server may be replaced. Relevant for checking trusted path/channels.	
Exposing functionality requiring authorization	Testing Approach(es): design	This pattern addresses how to check that the system disallows a user to action an object if she has not the proper credentials.	
Sensitive information confidentiality	Testing Approach(es): architectural	This pattern addresses how to check that the system use a known safe encryption protocol. Relevant to test if any sensitive or personal information contained within an object is only accessible to the user who acted it.	

5.2 SECURITY TEST PATTERN APPLICATION IN THE CASE STUDIES

As above mentioned, several security test patterns have been defined covering several security areas. One of our goals was therefore to study their applicability to the WP1 industrial use cases. In the following, we present the patterns applied (by FOKUS, IT and INPG) to three different case studies provided by the industrial DIAMONDS partners, focused on three different industrial domains: the Banking, Automotive and Smart cards areas.

5.2.1 Banking case study: Giesecke & Devrient (FOKUS)

The testing process in the Giesecke & Devrient case study has started with a concise security risk analysis following the CORAS methodology. As result of the risk analysis, several vulnerabilities were revealed that should be tested if they actually exist within the SUT. In order to generate appropriate tests for these vulnerabilities, the application of security test patterns has been considered a suitable way to select test generation techniques and test procedures. Those security test patterns constitute the link between security risk analysis and security testing. In the case study two security test patterns are fitting to the results of the risk analysis and have been applied to the test generation process of the Giesecke & Devrient case study.

1. Security test pattern: **Usage of unusual behavior sequences**. The application of this security test pattern leads to behavioral fuzzing in order to generate attacks based on invalid message sequences. The test system creates invalid message sequences instead of invalid input data by modifying functional test cases. The behavioral fuzzing approach developed in DIAMONDS uses UML sequence diagrams and modifies these. The generated test cases aim at revealing authentication bypass vulnerabilities by submitting messages for configuring the banknote processing system before or without authentication.
2. Security test pattern: **Detection of vulnerability to injection attacks**. The application of this security test pattern leads to data fuzzing approaches (e.g. to detect SQL injection flaws) were applied by a new developed fuzz testing extension for TTCN-3. Data fuzzing sends a large number of invalid values to the system under test at certain points within a test case

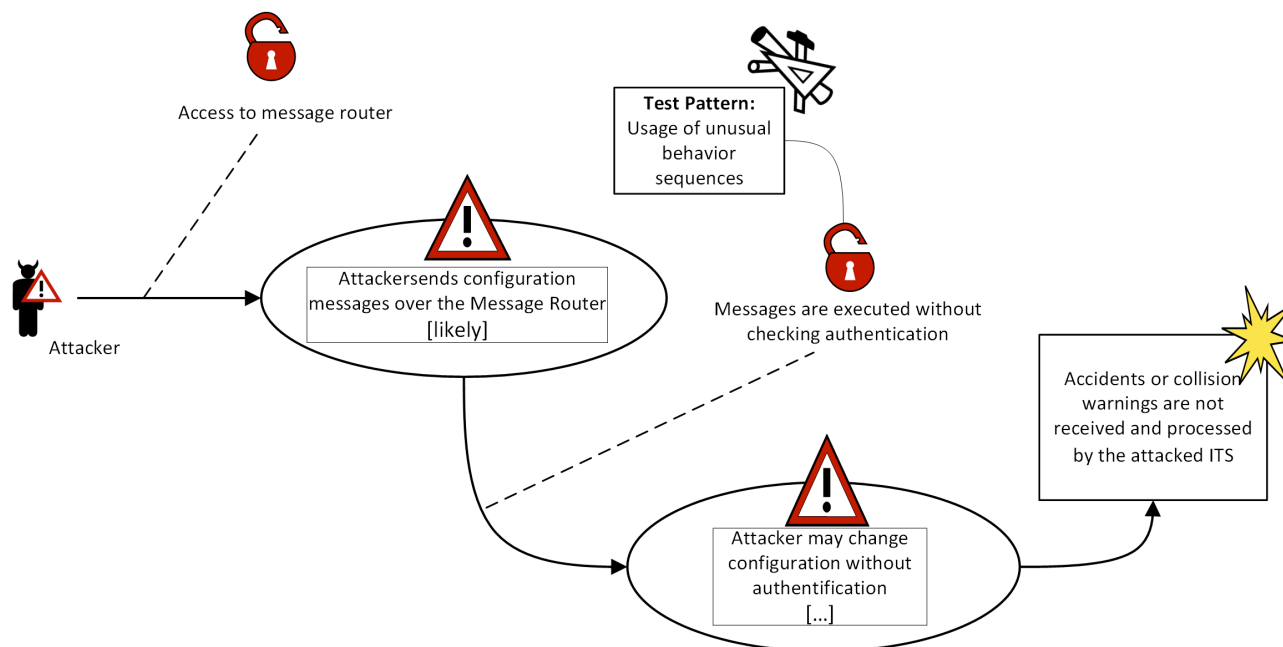


Figure 18: Application of test pattern to risk assessment artefacts


The application of test pattern allows for an early identification of test techniques and test methods. They can be applied directly after the initial security risk assessment has been completed. In DIAMONDS we have developed a tool supported traceability method that allow relating artefacts from the security risk assessment with artefacts from the testing and preserving this relationship over the whole (test) development process. Following this approach the assignment of test pattern to risk assessment artefacts is one of the first steps for testing. Figure 18 shows the assignment of the test pattern "Usage of unusual behavior sequences" to one of the risk assessment models of the G&D case study. On basis of the test pattern and its assignments to risk assessment artefacts we have started the test specification and generation following the recommendations given by the test pattern.

5.2.2 Automotive case study: Dornier Consulting (IT)

Security test pattern: *Usage of Unusual Behavior Sequences*

The test procedure template provided in D3.WP4.T1 for this particular test pattern was mainly intended for active testing. However, for our passive testing approach, we have tried to adapt certain steps from the template and modified some parts of it to suit our testing approach. Based on the template provided for the security test pattern, the following steps were followed to detect the attack pattern.

- 1) Based on the information obtained from the case study provider (DCo) and the Bluetooth specification the Bluetooth pairing scenario between the car's audio system and the Bluetooth enabled phone was modeled using the sequence diagram to understand the sequence behavior.
- 2) As our approach was based on passive testing or post-mortem analysis, we collect the trace from the system and observe if the normal behavioral sequence was correctly carried out.
- 3) According to the template the system was subjected to each of the new behavior sequences observed and for each of those the system was analyzed for any exceptional behavior. But in our approach in order to create an exceptional behavior we introduced an attack scenario called Bluestabbing attack manually in the trace obtained to differentiate it from the normal behavior. In the

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 37 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

Bluestabbing attack, the attacker impersonates as a legitimate user and modifies the Bluetooth device name of a legitimate user by resending a message with a badly formatted device name by causing the slave device to confuse during the device discovery phase (Inquiry). But this attack could be more severe, if the Bluetooth attacker modifies his own local device name as a legitimate user's device name, and tries to establish a connection with the other Bluetooth device by capturing the passwords and sensitive information from the device (a detailed description is given in D5.WP2).

- 4) The system trace with the attack scenario was again monitored using our symbolic passive testing approach as described in the D5.WP2 to detect the behavioral and attack sequence.
- 5) The prototype model at the end of the evaluation provides a verdict Pass/Fail/Attack-Pass/Inconclusive.

5.2.3 Smart cards case study: Gemalto (INPG)

Security Test Pattern: *Detection of Vulnerability to Injection Attacks*


Based on the template described in D3WP4T1 in section 4.1.7 "Detection of Vulnerability to Injection Attacks", INPG incorporated various steps from the pattern in the proposed method that automates the process of generating inputs that trigger XSS vulnerability in a web application. INPG has adopted, adapted and formalized the pattern and following is the short description of applying it on Gemalto case study.

Grenoble INP searched for Cross Site Scripting (XSS) vulnerabilities in the Gemalto case study. XSS is a particular instance of the command injection problem, in which an attacker sends input values crafted to escape outside a structure in which the developers assumed the input would be confined. By doing so, attacker can make the SUT to execute code they control and which was not intended by the developers. In the case of XSS, the objective is to escape outside an HTML parse tree node. XSS patterns are characterized by a taint flow from an input parameter (e.g., variable in an HTTP GET request) to an output (HTML webpage). Detecting such patterns is not trivial as it requires the tester to be able to navigate in the SUT and to exert a wide range of input values, thus it requires knowledge about the control flow and the data flow of the SUT.

Grenoble INP created an Evolutionary Fuzzing approach (see doc. D3WP2), implemented in a tool KameleonFuzz, that automatically infers the necessary knowledge to focus on parts of the SUT on which this taint flow property hold. Once this knowledge is acquired, KameleonFuzz exerts a wide range of "fuzzed" (malicious) values in input and observes whether an XSS pattern is observed or not. Such inputs are evolved towards better ones according to a heuristic (XSS specific) to increase the chance of discovering XSS. *In effect, Genetic Algorithm (GA) automates the process of generating inputs that contain certain "malicious data" as mentioned in the security test pattern template.* These inputs have more chances of triggering XSS vulnerability.

However, due to limited access, Grenoble INP could not test it with full coverage of the SUT (i.e. TSM). As a result, no XSS vulnerabilities were found. It was also noticed that the application (SUT) was built with good care for such vulnerabilities.

For the same case study, Smartesting formalized test patterns through test purposes (cf D5.WP2 and D5.WP3 DIAMONDS deliverables) and use CertifyIt technology to generate attack vectors on Gemalto TSM.

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 38 of 39
		Version: 1.0
		Date : 16.05.2013
		Status : Final Confid : Public

6. CONCLUSION


This document constitutes the third and final deliverable of work package 4, documenting results of task T4.1 (security patterns) and tasks 4.2 and task 4.3 on risk- and model-based security testing methodologies.

While the other work packages of the DIAMONDS project describe techniques/methods and tools, work package 4 describes processes/guidelines for applying these tool and techniques in practice.

The main objective of this deliverable has been to define a generic process for model-based security testing, provide examples of concrete instances of the process, and to provide an overview of the security test patterns that have been identified in the DIAMONDS project.

The generic process for model-based security testing combines four main areas: testing, model-based testing, security testing, and risk assessment. At the beginning of this document (in Section 1), we provided representative processes for each of the four areas based on existing standards and guidelines. Then in, Section 2, we combined these processes into a generic process for model-based security testing, and gave an overview of how the DIAMONDS techniques relate to the activities of the process.

In Section 3 and Section 4, we presented concrete instances/refinements of the generic process, focusing on model-based security testing and test-based security risk assessment, respectively. Finally, in Section 5, we provided an overview of the security test patterns that have been identified in the DIAMONDS project, and described their application in the DIAMONDS case studies

	DIAMONDS Security Testing Methodology Deliverable ID: D5_4_T1-T3	Page : 39 of 39
		Version: 1.0 Date : 16.05.2013
		Status : Final Confid : Public

7. REFERENCES

- [1] Information security. http://en.wikipedia.org/wiki/Information_security, last date accessed 17.09.2011.
- [2] Model based testing. http://en.wikipedia.org/wiki/Model-based_testing, last date accessed 17.09.2011.
- [3] P. Bourque and R. Dupuis. Guide to the software engineering body of knowledge 2004 version. Technical report 19759, IEEE Computer Society, 2004.
- [4] The Committee on National Security Systems. National Information Assurance (IA) Glossary, CNSS Instruction No. 4009, 2010.
- [5] IEEE Computer Society. IEEE 829 - Standard for Software and System Test Documentation, 2008.
- [6] International Standards Organization. ISO 27000:2009(E), Information technology - Security techniques - Information security management systems - Overview and vocabulary, 2009.
- [7] International Standards Organization. ISO 31000:2009(E), Risk management- Principles and guidelines, 2009.
- [8] International Standards Organization. ISO 29119 Software and system engineering - Software Testing-Part 2 : Test process (draft), 2012.
- [9] M.S. Lund, B. Solhaug, and K. Stølen. Model-Driven Risk Analysis: The CORAS Approach. Springer, 2011.
- [10] The Open Group. The Open Group Architecture Framework Version 9.1, 2011.
- [11] Testing Standards Working Party. BS 7925-1 Vocabulary of terms in software testing. 1998.
- [12] F. John Reh. Glossary of business management terms and abbreviations. <http://management.about.com/cs/generalmanagement/g/objective.htm>, last date accessed 19.04.2012.
- [13] S. Chao William. System Analysis and Design: SBC Software Architecture in Practice. Lambert Academic Publishing, 2009.
- [14] Scarfone K., M. Souppaya, A. Cody, A. Orebaugh: Technical Guide to Information Security Testing and Assessment. NIST Special Publication 800-115 (2008)
- [15] ETSI (European Telecommunication Standards Institute): Methods for Testing & Specification (MTS); Model-Based Testing (MBT); Requirements for Modelling Notations, ES 202 951 v 1.1.1 (2011)
- [16] R. Binder. Testing Object Oriented Systems: Models, Patterns and Tools. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1999.
- [17] A. Vouffo Feudjio, I. Schieferdecker; Test patterns with TTCN-3; Proceedings from the International Workshop on Formal Approaches to Testing of Software - FATES/RV , pp. 170-179, 2004
- [18] C. Alexander; A Pattern Language: Town, Buildings, Construction; Oxford, UK: Oxford University Press, 1977
- [19] Department of Defense: Proceedings for Performing a Failure Mode, Effects and Criticality Analysis, MIL-STD-1629, Washington 1949/1980, <http://www.fmea-fmeca.com/milstd1629.pdf> (2012-04-15)
- [20] Joanne Bechta Dugan, Kevin J. Sullivan, David Coppit: Developing a low-cost high-quality software tool for dynamic fault-tree analysis, IEEE Transactions on Reliability 2000-03 pp. 49-59, IEEE Computer Society 2000, ISSN: 0018-9529, Digital Object Identifier: 10.1109/24.855536
- [21] R. Binder. Testing Object Oriented Systems: Models, Patterns and Tools. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1999.
- [22] A. Vouffo Feudjio, I. Schieferdecker; Test patterns with TTCN-3; Proceedings from the International Workshop on Formal Approaches to Testing of Software - FATES/RV , pp. 170-179, 2004
- [23] G. Erdogan, Y. Li, R. K. Runde, F. Seehusen, K. Stølen: Conceptual Framework for the DIAMONDS Project, Oslo May 2012
- [24] Ida Hogganvik, Ketil Stølen: A Graphical Approach to Risk Identification, Motivated by Empirical Investigations, 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2006), Lecture Notes in Computer Science 4199 pp. 574-588, Springer Berlin Heidelberg 2006, DOI: 10.1007/11880240_40
- [25] B. Smith, L. Williams: On the Effective Use of Security Test Patterns, Proceedings of the Sixth International Conference on Software Security and Reliability (SERE), 2012 IEEE, pp. 108-117, DOI: 10.1109/SERE.2012.23