Compositional Risk
Assessment and Security
Testing of Networked Systems

# Deliverable D3.2.2

# Techniques for Compositional Test-Based Security Risk Assessment v.2

RASEN - 316853

| | |
|---|---|
| **Project title:** | RASEN |
| **Project number:** | 316853 |
| **Call identifier:** | FP7-ICT-2011-8 |
| **Objective:** | ICT-8-1.4 Trustworthy ICT |
| **Funding scheme:** | STREP – Small or medium scale focused research project |

| | |
|---|---|
| **Work package:** | WP3 |
| **Deliverable number:** | D3.2.2 |
| **Nature of deliverable:** | Report |
| **Dissemination level:** | PU |
| **Internal version number:** | 1.0 |
| **Contractual delivery date:** | 2014-09-30 |
| **Actual delivery date:** | 2014-09-30 |
| **Responsible partner:** | Software AG |

## Contributors

| | |
|---|---|
| Editor(s) | Bjørnar Solhaug (SINTEF) |
| Contributor(s) | Bjørnar Solhaug (SINTEF), Ketil Stølen (SINTEF), Johannes Viehmann (Fraunhofer), Frank Werner (Software AG) |
| Quality assuror(s) | Erlend Eilertsen (Evry), Samson Esayas (UiO) |

## Version history

| Version | Date | Description |
|---|---|---|
| 0.1 | 14-07-08 | Table of contents and document outline |
| 0.2 | 14-08-14 | First draft of all sections |
| 0.3 | 14-09-08 | Finalized for internal review |
| 0.4 | 14-09-26 | Revision after internal review |
| 1.0 | 14-09-30 | Quality checked and finalized |

## Abstract

This deliverable reports on the main results of RASEN WP3 from the second year of the project. The tasks that have been addressed are: (T3.1) the development of techniques for compositional security risk assessment, (T3.2) the development of techniques for test-based security risk assessment, and (T3.3) the development of techniques for continuous risk assessment by means of test-based indicators.
The RASEN approach to compositional security risk assessment has been further developed, and this deliverable introduces our notion of risk model encapsulation. We have developed modeling support for composing individual risk models, where the encapsulation allows the models to be combined without having to consider or assess the internal details of the respective models. The techniques and tools for test-based security risk assessment have been extended in several directions. The deliverable presents results covering (semi-) automated risk modeling, security testing, and security test result aggregation. The deliverable finally presents techniques for continuous security risk assessment by monitoring and aggregation of key indicator values, where the indicators provide information about the current risk picture at any point in time.

## Keywords

Security, security risk assessment, test-based security risk assessment, compositional security risk assessment, risk monitoring, risk modeling, risk assessment tools.

# Executive Summary

The overall objective of RASEN WP3 is to develop tools and techniques to facilitate compositional security risk assessment supported by security testing. This includes developing tools and techniques i) for compositional security risk assessment and security testing, ii) for identifying, estimating and verifying security risks based on security test results, and iii) for reuse of risk assessment and security test results, as well as dynamic updates of the security risk assessment based on test results.

This deliverable reports on the WP3 results after the first year of the project. The results cover all of the WP3 research tasks, namely (T3.1) the development of techniques for compositional security risk assessment, (T3.2) the development of techniques for test-based risk identification and estimation in order to complement the risk picture based on test results, and (T3.3) the development of techniques for continuous risk assessment of large scale systems by the use of test-based indicators. In particular, the deliverable makes the following contributions.

- A tool-supported approach to risk modeling and assessment of large-scale networked systems. The risk modeling involves the automated or manual assignment of vulnerabilities to components, where vulnerabilities can be imported from existing catalogues or databases. The approach is integrated into the RASEN methodology which allows the export of components to the security test execution.
- Techniques for the systematic use of security test results to update the risk picture. In particular, we explain how we make use of test measures and test metrics to capture the results of the security testing. The low-level test measures and metrics are in turn aggregated into more high-level risk measures and metrics that serve as the input to the risk assessment.
- A tool-supported method for the combination of test-based risk assessment and risk-based security testing. The tool is designed to support the RACOMAT (Risk Assessment COMbined with Automated Testing) method for risk assessment and modeling, test procedure identification, as well as test execution and incident simulation.
- Techniques and modeling support for compositional security risk modeling and assessment. At the core of the approach is a notion of risk model encapsulation with modeling techniques for hiding the internal details of each individual risk assessment. The risk model encapsulation supports the composition of risk models by considering only the information that is visible on the defined interface of the encapsulated models.
- An approach to continuous security risk assessment by means of risk monitoring. The risk monitoring is enabled by means of the monitoring of key indicators, where the indicators provide information about the current risk picture at any point in time. We explain how key indicator values can be aggregated to derive risk information.

The WP3 results contribute to support and facilitate the overall RASEN methodology that is presented in the context of WP5. The WP3 tools are moreover being integrated into the RASEN tool-box and have therefore the potential to be used in combination with other RASEN tools and techniques.

# Table of contents

# 1 Introduction

A main objective of RASEN WP3 is to develop techniques and tools that facilitate security risk assessment of large-scale and complex software systems. To fulfill this objective we are conducting R&D activities in three directions. First, we are developing techniques and modeling support for compositional security risk modeling and assessment. Such techniques should allow large system to be decomposed into smaller sub-systems or components that can be analyzed separately. For this we need methods for deducing the combined results of the individual analyses. Second, we are developing techniques for test-based risk identification and estimation, so as to complement the risk picture based on the test results. Third, we are investigating techniques for continuous security risk assessment by leveraging the techniques for compositional security risk assessment, and by means of risk monitoring.

The current WP3 status and the second year results of these R&D activities are presented in this deliverable. The activities correspond to research tasks T3.1 (compositional security risk assessment), T3.2 (test-based risk identification and estimation) and T3.3 (continuous risk assessment) respectively, of RASEN WP3. More specifically, the technical contents of this deliverable are as follows.

Section 2 presents the ARIS tool-supported approach to risk modeling and assessment of large-scale networked systems. Using this approach, a software product is specified by decomposing it into components that in turn are further decomposed to form a tree structure of any depth. The risk modeling involves the automated or manual assignment of vulnerabilities to components, where vulnerabilities can be imported from existing catalogues or databases. The approach is integrated into the RASEN methodology which allows the export of components to the security test execution.

Section 3 concerns the use of the security test results to update the risk picture. In particular, we explain how we make use of test measures and test metrics to capture the results of the security testing. The low-level test measures and metrics are in turn aggregated into more high-level risk measures and metrics that serve as the input to the risk assessment.

Section 4 presents tool-support for the combination of test-based risk assessment and risk-based security testing. The tool is designed to support the RACOMAT (Risk Assessment COMbined with Automated Testing) method for risk assessment and modeling, test procedure identification, as well as test execution and incident simulation. The tool is being integrated into the RASEN toolbox, and the RACOMAT method can be used in combination with other RASEN methods and techniques to support the overall RASEN methodology.

Section 5 presents an approach to compositional security risk modeling and assessment. The purpose of the approach is to allow large systems to be decomposed into smaller parts that are assessed and analyzed independently. The individual results can then later be composed to form the risk model and risk assessment results for the system as a whole. At the core of the approach is a notion of risk model encapsulation with modeling techniques for hiding the internal details of each individual risk assessment. The risk model encapsulation supports the composition of risk models by considering only the information that is visible on the defined interface of the encapsulated models.

Section 6 presents an approach to continuous security risk assessment by means of risk monitoring. The risk monitoring is enabled by means of the monitoring of key indicators, where the indicators provide information about the current risk picture at any point in time. Such security risk monitoring is sometimes referred to as passive testing, whereas the security testing described in the previous sections and in RASEN WP4 are referred to as active testing. When conducting test-based security risk assessment, there is in both cases a need for aggregating the test results. In this section we explain how the aggregation can be done by the aggregation of key indicator values.

The presented methods and techniques support various parts of the overall RASEN methodology as presented in the context of WP5. The WP3 tools are moreover being integrated into the RASEN toolbox, which means that they can be used in combination with other RASEN tools. The current WP3 tools are provided in prototype deliverable D3.3.2.

# 2 Risk Modeling of Large Networked Systems

The following chapter discusses the design of a modeling framework on top of the RASEN methodology to support risk assessment and modeling of large scale networked software systems. In this sense, we present a component type representation which allows a high degree of freedom when modeling a software component, and an export/import mechanism to interface the RASEN testing framework.

The existing risk template has been extended to be self-documented which greatly facilitates the modeling of security risks. In a second step, generic templates are generated that allow a convenient modeling by the end-user offering drag-and-drop techniques where available risk templates can be reused for the security risk assessment of new software products. With this implementation we accomplished an automated generation of this risk templates based on the most common security risk templates and industry standards like the Common Weakness Enumeration (CWE) database [10].

In the next step the existing model has been extended to express links between currently defined risks and weaknesses on the component level. The Common Weakness Scoring System (CWSS) [11] provides hereby a mechanism for scoring weaknesses in a consistent, flexible and open manner while accommodating context for various business domains. As it is part of the CWE project, it can be interlinked with information already available in the models to allow quantitative measures of available weaknesses present within a software component.

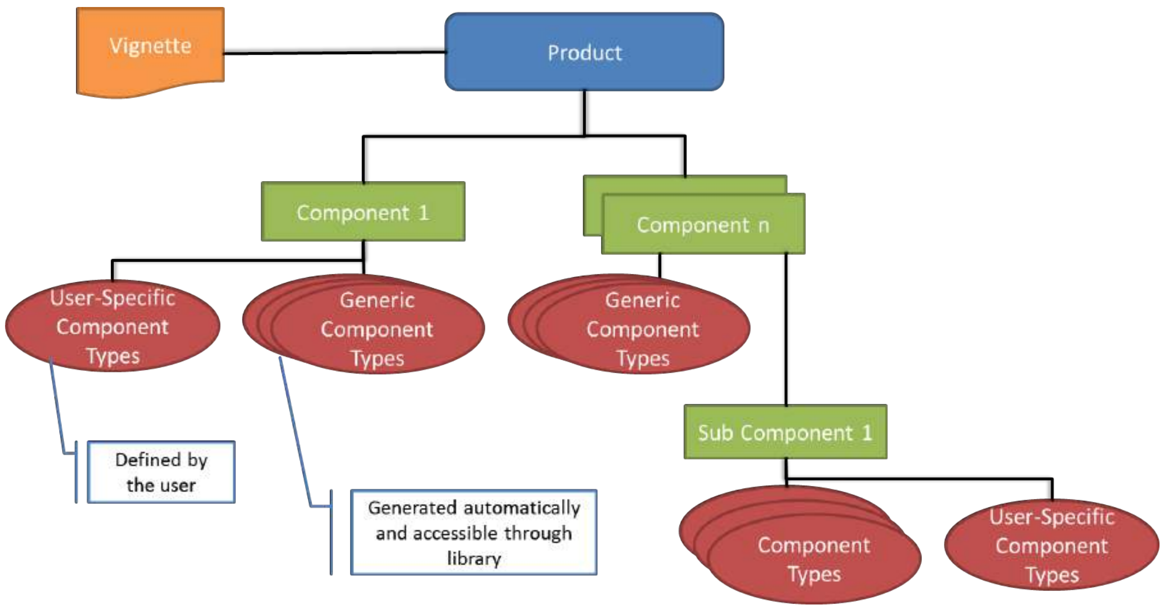


**Figure 1 – Model overview of the product, component and component types**

The current model capabilities are depicted in Figure 1. Starting from the top most component, each product is assigned a vignette which defines the application area in which the product is used. This could be for example an Internet-hosted system with a high degree of networking as it is found in cloud computing, but also as simple as a system deployment in a company's infrastructure with maximum security and intrusion protection through multi-layer firewalls.

## 2.1 Component Type Implementation and Assignment

In the RASEN ARIS model each product consists of components which build the product hierarchy and can be interleaving in any thinkable form by forming sub-components of arbitrary depth. Each component is assigned component types specifying the core functionality and nature of the components' application like networking, Linux Operation System, Authentication, etc. These components have two different origins: On the one hand, they can be taken from a generated library of Generic Component Types, and on the other hand, they can be user-defined component types where users individually assess the type and the weaknesses which this component type exhibits. This distinction is illustrated in Figure 2. The two different component types are described in detail in the following two sections.

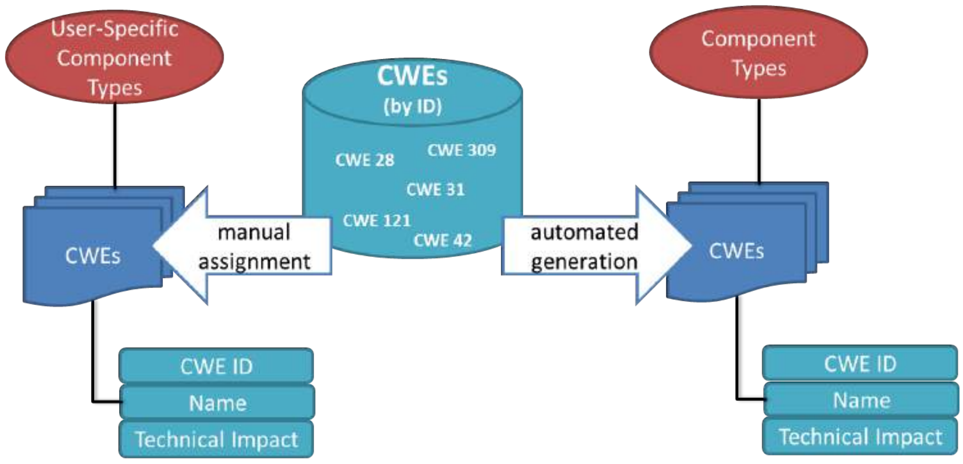


**Figure 2 – Different natures of component type libraries**

### 2.1.1 Generic Component Type Generation

The idea behind the generic component type generation is to support software developers and risk experts by providing generic modules for an improved and much faster risk assessment. In this sense, it is easier and less error-prone to choose from a list of best-known weaknesses as to choose the mode time in which the weaknesses for each individual component have to be assessed and derived from scratch. The concept is shown on the right side of Figure 2.

The list of generic component types is generated according to best practice and follows the list of weaknesses derived from the Common Weakness Enumeration (CWE) schema. The CWE database is an array of CWE-IDs which either represent a weakness, a category, a view or a compound element. Views and compound elements are ignored as they are not important for the algorithm. A category contains one or more weaknesses and can also contain one or more child categories. A weakness can be a weakness class (this is also a kind of category), a base weakness or a weakness variant. Weaknesses can be in a “Can-also-be”-relationship with another weakness. So the CWE database can be thought of as a tree whose leafs can be connected with each other. The following graphic clearly visualizes the dependency from categories to multiple weaknesses derived from the CWE database.

For most of the weaknesses the following data is available: Title and short description, applicable platforms (this can be an operating system, framework or a programming language), technical impact, affected resources, functional area and consequence scope.

For each component type, the weaknesses (CWE-IDs) are assigned using one or more of the following methods following an automated generation process:

1. Collecting all CWE-IDs of a CWE category: For the specified category, weaknesses are collected recursively following the tree structure.
2. Searching for CWE-IDs using a search term: CWE weaknesses have a headline and a short summary description. All CWE weaknesses that contain the search term are added to the component type.
3. Collecting all CWE-IDs which share a specified value. A shared value can, for example, be operating system, framework, programming language, etc.
4. Adding CWE-IDs manually: Some CWE-IDs are added manually to component types. This is necessary because not all CWE-IDs can be assigned automatically using one of the methods described above. The most important reasons for that is incomplete and/or inconsistent data in the CWE database.

There is also a component type called “Generic” which contains all CWE weaknesses that are too generic to fit to one of the other component types. The component types listed in Table 1 are automatically generated and allow the security experts and software architects quick modeling of their software systems.

| Generic Component Type Name | N° of CWEs assigned | Generic Component Type Name | N° of CWEs assigned |
|---|---|---|---|
| Core | 176 | Transport Protocols **HTTP** | 9 |
| Using APIs | 16 | Transport Protocols **FTP** | 1 |
| Multithreading | 31 | Operating Systems Windows | 10 |
| Database Access | 10 | Operating Systems Mac OS | 2 |
| File System Access | 52 | Operating Systems UNIX/Linux | 4 |
| Reading and Writing Files | 10 | Framework Struts | 10 |
| Logging | 5 | Framework.NET | 6 |
| Temp Files | 3 | Framework J2EE | 9 |
| Configuration | 20 | Programming Language Java | 70 |
| Authentication | 28 | Programming Language C | 70 |
| Authorization | 12 | Programming Language C++ | 70 |
| Using Cryptography | 19 | Programming Language Assembly | 4 |
| Using Random Values | 20 | Programming Language Ruby | 3 |
| Sensitive Data | 9 | Programming Language Python | 3 |
| Networking | 2 | Programming Language C# | 3 |
| Critical Resource Access | 82 | Programming Language ASP.NET | 7 |
| Server | 4 | Programming Language PHP | 13 |
| Web Application | 143 | Programming Language JavaScript | 1 |
| Mobile Application | 26 | Programming Language XML | 6 |
| Graphical User Interface | 17 | Generic | 117 |
| User Controlled Input | 22 | | |

**Table 1 – Generated generic component types**

### 2.1.2 User-Specific Component-Type Definitions

As in some cases the automatically generated component type is not properly fitting in the component concept, there is an additional way of manually creating and defining component types as shown on the left side of Figure 2.

In this step, the user can create individual component types and manually assign CWEs from the CWE database to this component. This type creation gives the security engineer full control over its component type definitions, and allows technically any possible component type declaration.

## 2.2 Interfacing the Testing Framework

The testing framework is interfaced using import and export facilities from the ARIS framework. From this point, the integration of the RASEN models is taking advantage of other tools developed within the project, e.g., the testing framework. Figure 3 shows how import and export interfaces are aligned with the Test Execution framework.

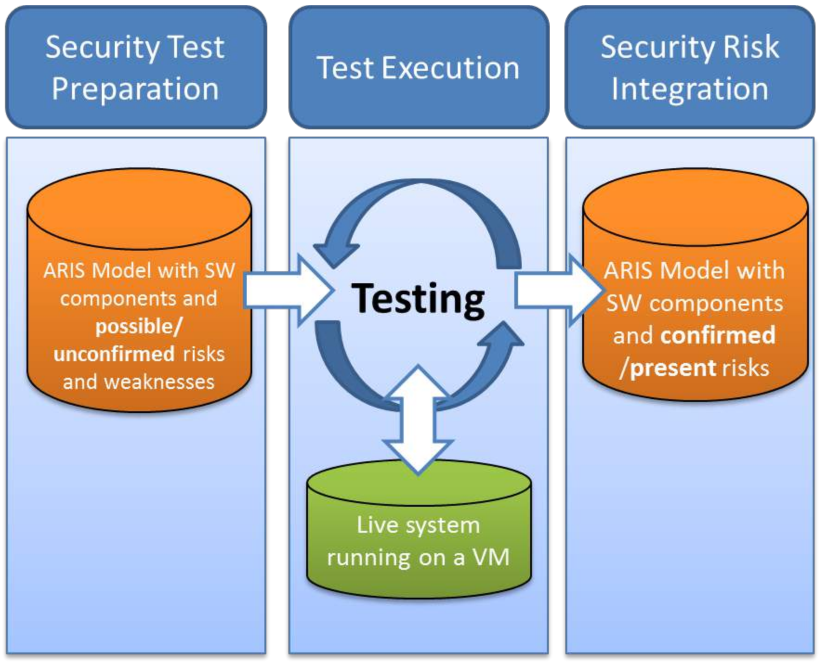

Security Test Preparation
Test Execution
Security Risk Integration
ARIS Model with SW components and possible/ unconfirmed risks and weaknesses
Testing
ARIS Model with SW components and confirmed /present risks
Live system running on a VM

**Figure 3 – Interfaces for import and export**

The import and export interface exchanging RASEN models with test and test generation are provided by export and import scripts that allow automated exchange of the relevant security assessment artifacts. This means in detail that the Security Tests are prepared for export by converting relevant RASEN artifacts from the ARIS model into an interchange format. From the security perspective, this export contains a list of possible and yet unconfirmed risks and weaknesses in the security risk model.

As an example, Figure 4 shows the RASEN model of Software AG's product called Command Central (CCE). In this figure, the top product is shown as CCE with the CCE Vignette describing the deployment scenario the list of component with the CWEs underneath. The related export of this model, acting as an interface to the test execution environment, is depicted in Figure 5.

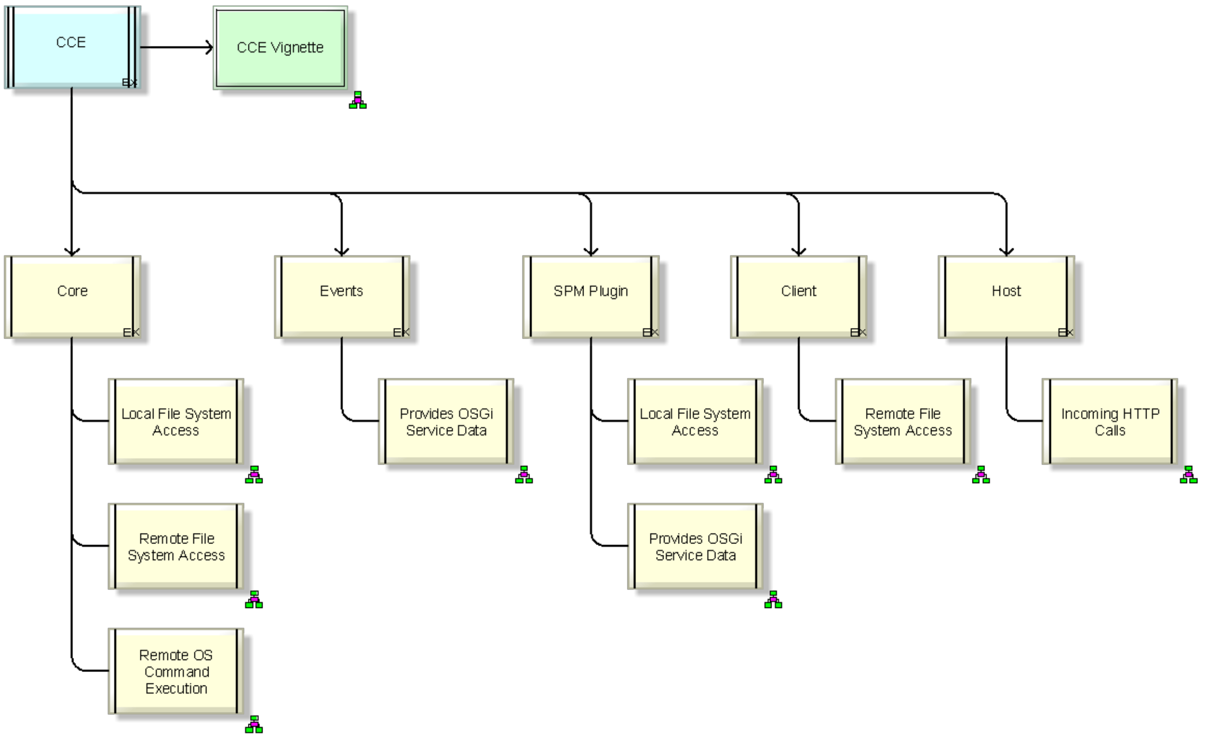

CCE
CCE Vignette
Core
Events
SPM Plugin
Client
Host
Local File System Access
Provides OSGi Service Data
Local File System Access
Remote File System Access
Incoming HTTP Calls
Remote File System Access
Provides OSGi Service Data
Remote OS Command Execution

**Figure 4 – Model of the Command Central Component with its RASEN model in ARIS**

```
#CCE Component
Product Name: CCE {
   "productName": "CCE",
   "components": [{
      "componentName": "Client",
      "subComponents": [],
      "cweList": [12, 641, 638, 646 ]
   }, {
      "componentName": "SPM Plugin",
      "subComponents": [],
      "cweList": [598, 588, 591, 582, 587,
113, 151, 149, 154, 116, 114, 113, 154,
140, 115, 141, 155, 156, 146, 150, 144   ]
   }, {
      "componentName": "Host",
      "subComponents": [],
      "cweList": [533, 540, 534, 536, 537 ]
   }, {
      "componentName": "Events",
      "subComponents": [],
      "cweList": [598, 588, 591, 582, 587,
113    ]
   }, {
      "componentName": "Core",
      "subComponents": [],
      "cweList": [12, 641, 638, 646, 151,
149, 154, 116, 114, 113, 154, 140, 115,
141, 155, 156, 146, 150, 144, 113, 524,
523, 54, 534, 525    ]
   }]
}
```

**Figure 5 – The ARIS export definition related to the CCE model in JSON format**

As illustrated in the figure above, the model export is realized by providing a file in JSON (JavaScript Object Notation) format representing the hierarchical structure of the software components and the list of assigned CWE entries from the CWE database.

After the Test Execution phase where weaknesses are tested on a life system, the found weaknesses are collected and grouped according to their components. Following up, the re-import of test results (that reveal the presence of weaknesses and risks) into the ARIS RASEN Database are provided and confirmed risks that are integrated into the risk picture.

This allows Software AG and other industry partners to apply the RASEN solution on small subsets of their software systems and enable security risk assessment not only on small subsets, but on large scale software systems as well.

## 2.3 Conclusion

Considering the risk modelling of large scaled networked systems, a valid solution should provide a high level of automation; human interaction may be prohibitively expensive considering the potential size and complexity of the software under consideration. The above mentioned approach relies on repositories like the CWE database, and provides a suitable means for modeling software systems while using already defined libraries to cluster functional aspects of the software into software components. It is further possible to group software components into functional clusters and reuse them in other software models to simplify the modeling process and enabling hierarchical models of technically any shape.

Following the RASEN approach, the integration of the model with the test execution framework is based on standard exchange formats i.e., JSON, to enable the exchange of information and provide an import and export interface. What is currently missing in the present implementation is the import of test execution results back into the ARIS RASEN model in order to aggregate the risk levels on component and product level. As this is considered as future work, this integration will be accomplished in the future implementation of the RASEN framework.

The presented techniques for software risk assessment are applied to the RASEN use case studies and evaluated against the requirements of the RASEN use case providers. This will eventually reveal the suitability of the presented approach in the area of industrial appliances and demonstrate the usability for each of the industrial RASEN partners.

# 3 Complement the Risk Picture Using Test Results

In this section we describe the RASEN techniques and modeling support for test-based risk assessment. With this approach we make systematic use of security test results to complement the risk picture and update the risk estimates and risk evaluation that is conducted as part of a security risk assessment. The focus of this section is on the aggregation of the security test results to derive the values that can be used to calculate and update the security risk assessment results.

The techniques build on the WP3 results presented in Section 6 and Section 7 of RASEN deliverable D3.2.1 [18] and is currently being applied and evaluated in the RASEN case studies. We refer to WP5 [15] for the detailed presentation of the RASEN method for test-based security risk assessment and to WP2 [19] for a presentation of the case studies where the method and techniques are applied.

## 3.1 Method Overview

Figure 6 illustrates our approach, focusing on the security risk assessment part. The risk analysis part is depicted to the left and follows the process of the ISO 31000 risk management standard [3]. After the context establishment, the risk assessment is conducted, which consists of risk identification, estimation and evaluation. After the risk evaluation the unacceptable risks are analyzed further for possible treatment.

The risk identification and estimation make use of expert judgments, historical data, statistics, catalogues of threats and vulnerabilities, etc. to document the risk picture and to estimate the likelihood of attacks and unwanted incidents. In many cases, however, such empirical data may be insufficient and thereby leave a certain degree of uncertainty of the risk assessment results. This uncertainty may, for example, be regarding the estimated likelihood of successful attacks, whether certain vulnerabilities exist, or how easy it is for attackers to exploit vulnerabilities.

The testing process is depicted to the right in Figure 6. From the perspective of the security risk assessment, the security testing is treated as a black box. This process receives the risk model from the risk assessment as input. The risk model serves as the basis for the test identification and prioritization. The results from the security testing are then fed back to the risk assessment. In the following we describe the steps in more details and explain how test data is aggregated and used to reduce the uncertainty of the security risk assessment results.

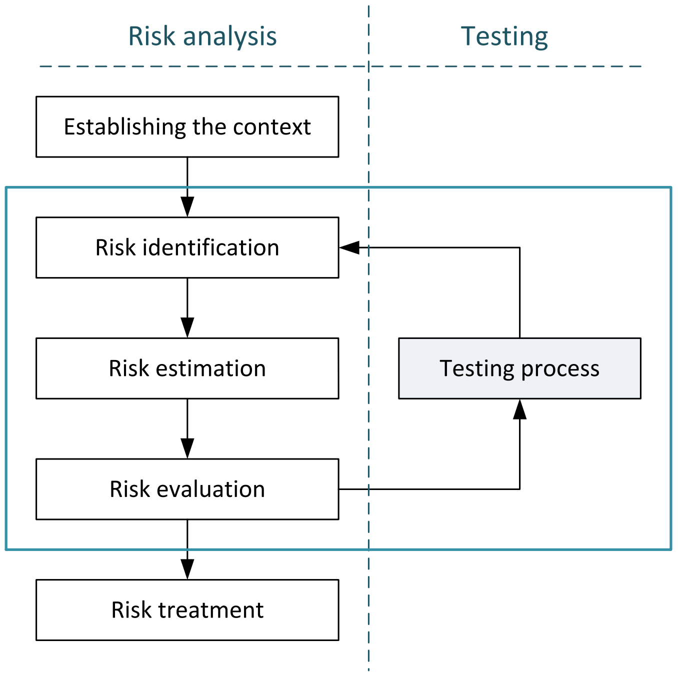


**Figure 6 – Test-based risk assessment**

## 3.2 Aggregating Test Results Using Risk Metrics

### 3.2.1 Concepts

In the RASEN approach we distinguish between measures and metrics. A measure is a specific piece of data, such as the known attack frequency on a specific vulnerability, whereas a metric is an aggregation or a set of measures. A metric is therefore more high-level than a measure, and is the value that is used as input by the security risk analyst or security tester. A metric is sometimes described as interpreted empirical data [6], such as the quantification of the degree of freedom from the possibility of an attack. A measurement, on the other hand, provides a single-point-in-time view of a specific discrete factor [6], where a security measurement is a particular value of an assessable security property of a system entity.

The concepts of measurements and metrics and how they are related are shown in Figure 7. The diagram is based on the RASEN data model [14], but somewhat adapted to focus on the conceptual aspects and on the risk assessment part. The diagram shows that a metric is a set of measurements, and that we have specializations of these concepts for both the security risk assessment and the security testing. A metric may have a type, and it can be assigned a value on a defined scale. A risk metric that is used as input to a security risk assessment is related to a risk model, which in turn includes a set of risk model elements. This means that a risk metric gives information about one or more such elements. Also the test measurements may be related to a set of risk model elements, but it is the aggregated test metrics that are applied by the risk analysts.

What is not explicitly shown in the diagram is the relation between test metrics and risk metrics. The test metrics aggregates test measurements, which are data that represent results from test procedures. The RASEN techniques for test-based security risk assessment make use of the test metrics to aggregate them into risk metrics. Such an aggregation is usually manual, but it can be conducted automatically for cases in which it is possible to specify and implement aggregation functions. In some cases we may also gather the test data and aggregate them on the level of measurements. The resulting risk measurements are then in turn aggregated into risk metrics.

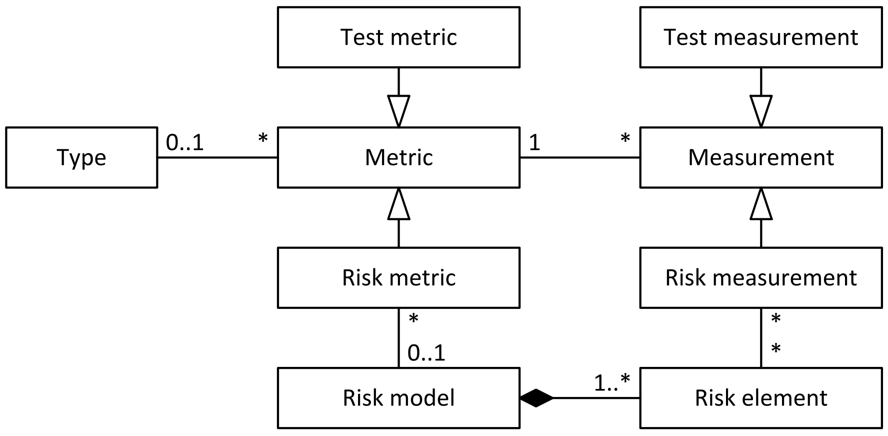


**Figure 7 – Measurements and metrics**

### 3.2.2 Process

In the following we describe the details of the process for aggregating the test results and applying them to update the risk models.

A risk model consists of elements and relations. Elements can, for example, be threats and unwanted incidents, whereas the relations are between the elements. Elements can have attributes, such as the likelihood and consequence of an unwanted incident. Relations can also have attributes, such as a vulnerability. We often assign likelihoods to relations, for example to specify the conditional probability that one incident may lead to another. These likelihoods can often be associated with the likelihood of existence of vulnerabilities, as well as the extent to which they can be exploited by an attacker.

In the presentation of the process we use a simple risk model notation that only shows threat scenarios (elements) and leads-to relations from one scenario to another. The relations are annotated with vulnerabilities. In Section 3.3 we give a concrete example extracted from a RASEN case study using the CORAS [7] notation.

We assume that a security risk assessment has been conducted and that we have a risk model with risk estimates that serves as the basis for the security testing. The process and the created documentation artifacts are as follows.

**Risk model.** The result of the risk assessment is a risk model with likelihood estimates as illustrated in Figure 8. The figure shows only a fragment of a model that consists of threat scenario *A* that may lead to threat scenario *B*, possibly due to the two vulnerabilities $V_1$ and $V_2$. The threat scenarios have been assigned the likelihoods $L_A$ and $L_B$, respectively, whereas the likelihood *L* on the relation is the estimated conditional likelihood for *A* to lead to *B*.

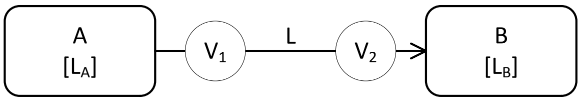


**Figure 8 – Risk model with estimates**

More often than not there is a degree of uncertainty of the likelihood estimates. The uncertainty should be explicitly specified, for example by assigning a degree of certainty to each estimate or by using likelihood intervals. In case we use intervals it is assumed that the correct likelihood is within the assigned interval.

The risk models with the estimates and the degree of uncertainty is used as the basis for the test identification, prioritization and selection. For a description of this particular task the reader is referred to WP4 deliverable D4.2.2 [17].

**Risk model and test procedure.** Figure 9 illustrates a test procedure identified on the basis of the risk model. A test procedure contains a sequence of test cases in their execution order. In this case the test procedure targets the relation from threat scenario *A* to threat scenario *B*, including the vulnerabilities. The test procedures typically include test cases regarding the identified vulnerabilities, but could also help identifying other vulnerabilities. Consequently the test results may give additional information about the conditional likelihood *L*, and therefore also the likelihood $L_B$.

Note that in the RASEN method the test procedure is linked to the test patterns, but this is something that is hidden from the risk assessment part which only makes use of the test results.

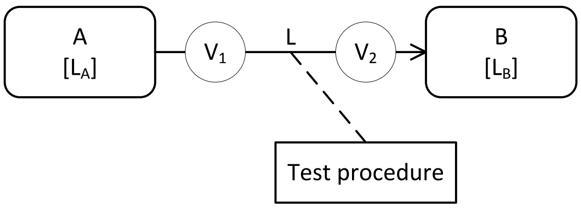


**Figure 9 – Risk-based test procedure**

**Test procedure and test measures.** The results of the testing can be represented by a set of test measurements as illustrated in Figure 10. These can in turn be aggregated into one or more test metrics, although this is not illustrated here. As for the test procedure, also this part is hidden from the risk assessment part. The reader is referred to D4.2.2 for details on the security testing.

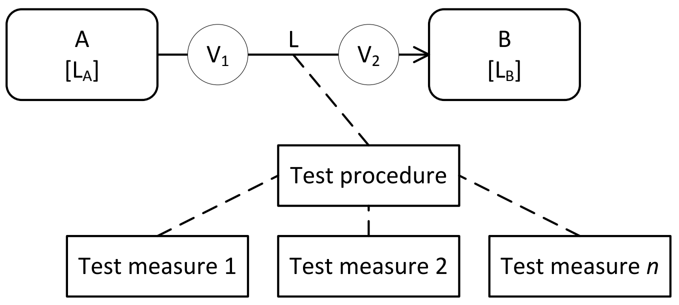


**Figure 10 – Test measures**

**Test procedure and risk measures.** Figure 11 illustrates the result of using the test measures (or metrics) to yield a set of risk measures (or metrics) that can be used as input to the risk assessment. Each set of risk measures is derived from a set of test measures, and therefore associated with a test procedure. This links the derived risk measures with the risk model elements that they provide information about.

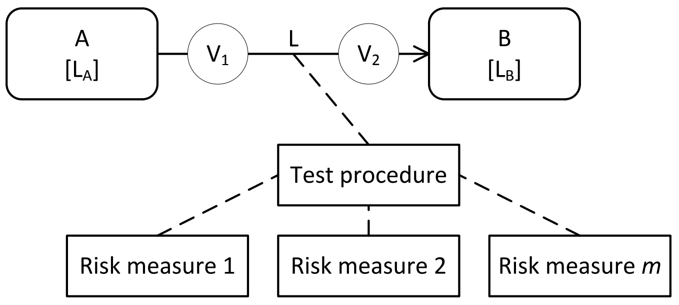


**Figure 11 – Test-based risk measures**

How to derive the risk measures from the test measures depends on the kind of measures. In some cases it would be possible to do an automated and tool-supported aggregation, but this depends on the type of measures and that we can specify and implement an aggregation function. For such tool support to be worth implementing, the specific testing and aggregation should be frequently reoccurring, otherwise it would be a one-off task each time. Alternatively the test results can be analyzed manually to derive the risk measures. Similar to a regular security risk assessment the test results are then used as part of the empirical data for the risk identification and estimation.

In the RASEN case studies we have derived the risk measures by expert judgments by risk analysts and personnel with insight into the target systems. By focusing on the vulnerabilities annotated on the relations from one threat scenario to another, we have in particular used measures on exploitability and on likelihood of existence. Exploitability is a measure of the degree to which a given vulnerability is easy to exploit by an attacker. Likelihood of existence is a measure of the likelihood that the vulnerability in questions is present in a system.

**Revised risk model.** In Figure 12 we illustrate the revised risk model after the aggregation of the risk measures. In this illustration there are two vulnerabilities, each of which typically yields a set of test cases. The test cases result in test measures that are aggregated to risk measures. Eventually the different sets of test measures must be aggregated to get a revised and hopefully improved estimate for the conditional likelihood *L* in Figure 8.

For the sake of the example let us assume that we have the measures *e* and *l* for exploitability and likelihood of existence, respectively, for the two vulnerabilities $V_1$ and $V_2$. Using these measures we investigate whether we can reduce the uncertainty of the likelihood *L*. If the uncertainty is captured by using a likelihood interval, our aim is to reduce the size of the interval *L*. Using the original estimate as input, together with the metrics, we derive the revised interval *L'* by a function *f* such that $L' = f(L,e,l)$. In some cases it may be possible to define the function formally, but in other cases we do the estimation of $L'$ by expert judgment. If $L' \subseteq L$ we have reduced the uncertainty and can update the risk model. If the uncertainty is not reduced we need to keep the original estimate.

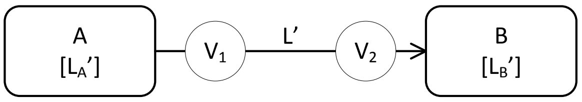


**Figure 12 – Revised risk model**

Because threat scenario *A* may lead to threat scenario *B*, the likelihood of the latter must be updated according to the change of *L* to *L'*. When the uncertainty of the conditional likelihood is reduced we get the updated estimate $L_B' \subseteq L_B$. The likelihood estimate of threat scenario *A* is not targeted by the illustrated security testing since it occurs before the tested relation and vulnerabilities. For illustration purposes, however, we have in Figure 12 replaced the likelihood estimate $L_A$ with $L_A'$. This is because the figure only shows a fragment of the risk model and that there may be other test cases that have been conducted and that give a basis for revising also this estimate. The estimation of $L_B'$ makes use of $L_A'$ in addition to $L'$.

## 3.3 Example

In the following we give a concrete example of the approach. The example is based on the InfoWorld case study, but note that we do not show here any actual incidents or real risk estimates for InfoWorld as this kind of information cannot be presented in an open, public report.

### 3.3.1 Context Establishment

**Target of analysis.** The target of the analysis is the Medipedia service (www.medipedia.ro). The service is provided as an application that can be accessed by clients via a web-interface. The analysis is limited to the attacks that can be performed via the interface, including by those that have a valid Medipedia account.

**Assets.** The assets that we consider in the presented example are *confidentiality of data* and *integrity of data*. The data is sensitive information such as personal data and medical data. The case study included also availability of service and compliance with regulations, but these are not shown in our example.

**Likelihood scale.** The likelihood scale that we use for the risk estimation is shown in Table 2. We use a scale of five values that correspond to consecutive intervals of frequencies.

| Likelihood | Definition | Interval |
|---|---|---|
| Seldom | Less than 1 time per 10 years | [0, 0.1>:1y |
| Unlikely | 1-10 times per 10 years | [0.1, 1>:1y |
| Possible | 2-12 times per year | [1, 13>:1y |
| Probable | 13-60 times per year | [13, 60>:1y |
| Certain | Over 60 times per year | [60, ∞>:1y |

**Table 2 – Likelihood scale**

**Consequence scale.** The consequence scale for each asset is defined similar to the likelihood scale, namely in terms of intervals of quantitative values. We do not need the precise definitions for the purposes of this section. It therefore suffices to give the names of the values, which as *insignificant*, *small*, *medium*, *high* and *critical*.

**Risk evaluation criteria.** We specified the risk evaluation criteria using the risk matrix. We divided the matrix into three parts, corresponding to the risk levels *low*, *medium* and *high*. The low risks are acceptable, whereas the high risk in principle are unacceptable and must be treated. The medium risks must be considered separately to decide whether they could be accepted or not.

### 3.3.2 Security Risk Assessment

In conducting the security risk identification and estimation we made systematic use of the Common Attack Pattern Enumeration and Classification (CAPEC) [12], as well as the Common Weakness Enumeration (CWE) [10]. We created risk models using the CORAS language by instantiating the relevant CAPEC attack patterns as CORAS threat diagrams.

This is exemplified in Figure 13 for the CAPEC-66 attack pattern that concerns SQL injection attacks. From the CAPEC attack pattern we get a description of how attackers can conduct the attack, a reference to relevant vulnerabilities from the CWE, the common likelihoods of such attack, the typical consequences and severity, and so forth. Note that in the example diagram we have not included the full CAPEC pattern for this attack.

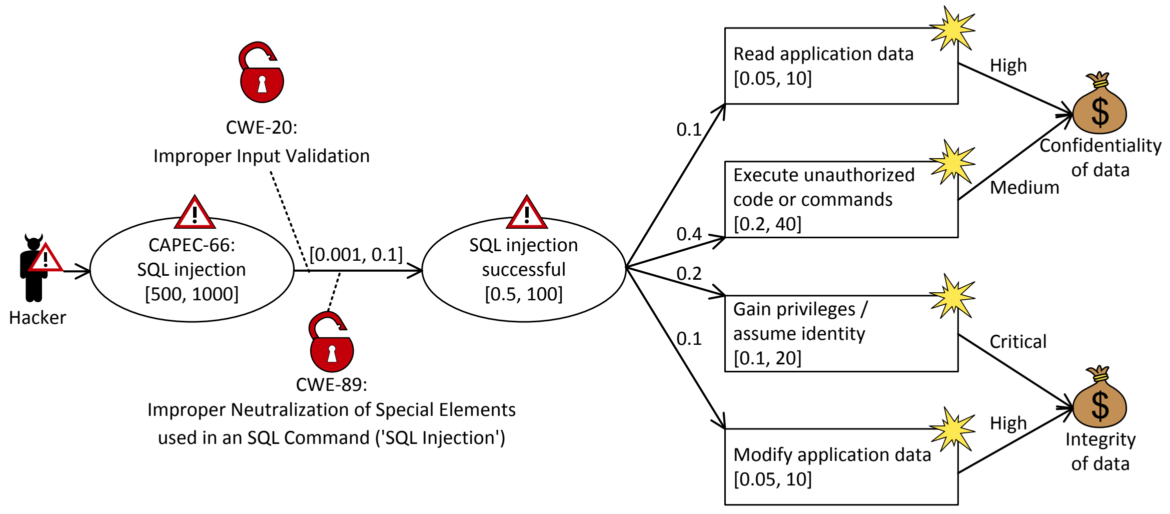


**Figure 13 – Threat diagram for the CAPEC-66 attack pattern**

The likelihood estimates shown in Figure 13 are only examples, and not the real estimates from the InfoWorld case study. In the real setting the estimates will be based on empirical data gathered from the client, such as InfoWorld, as well as expert judgments and the data provided by CAPEC. In the diagram we have used frequencies for the scenarios and incidents in terms of number of occurrences per year. For example, *CAPEC-66: SQL injection* is estimated to occur between 500 and 1000 times per year. We have used probabilities on the relations to specify conditional likelihoods. For example, the estimated probability that *CAPEC-66: SQL injection* will lead to *SQL injection successful* when the former occurs is in the interval between 0.001 and 0.1.

To do the risk evaluation we compare the identified risks and their risk levels with the risk evaluation criteria by plotting them into the risk matrix as shown in Figure 14. The widths of the likelihood intervals of the risk estimation reflect the uncertainty of the estimation. We see, for example, that the risk *Execute unauthorized code or command* span the three likelihood values of *unlikely*, *possible* and *probable*.

In some cases the uncertainty is not significant. This is in particular the case when the estimated risk level is within one risk level only. This is the case for the risk *Gain privileges / assume identity*, which is unacceptable no matter whether the likelihood is *unlikely*, *possible* or *probable*. However, when the uncertainty is such that we cannot determine whether the risk is acceptable or not, we may need to gather more evidence. In our case we do it by security testing.

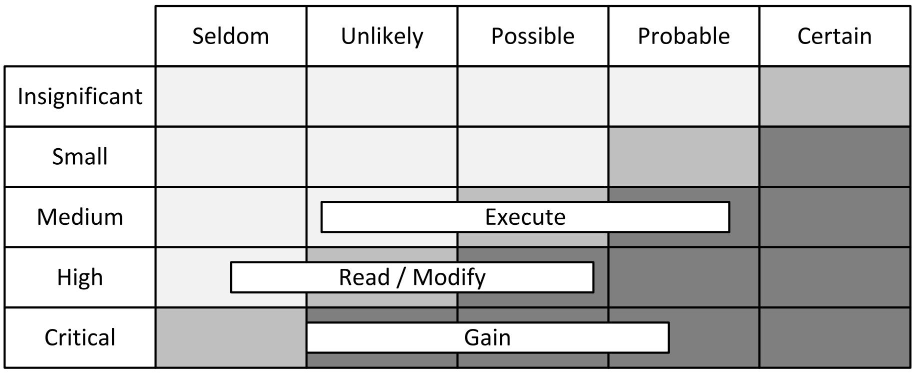


**Figure 14 – Risk evaluation**

### 3.3.3 Test Procedures and Test Measures

As the risk-based test identification and the security testing is not the concern of WP3, we refer to WP4 deliverables D4.2.1 [16] and D4.2.2 [17] for the details on that part of the RASEN process. From the viewpoint of the risk assessments we need as input the test measures resulting from the executed test procedures. As explained in D4.2.2 the test procedure derived from the example diagram in Figure 13 results in the following test procedure.

> Check that *CAPEC-66: SQL injection* leads to *SQL injection successful* with conditional likelihood *[0.001, 0.1]* due to *CWE-20: Improper Input Validation* and *CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')*.

For the sake of the example we assume here that the result of the conducted test cases based on the given test procedure provided two sets of test measures, one for each of the vulnerabilities. Each set consists of one measure for likelihood of existence (*l*) and one measure for exploitability (*e*). The likelihood of existence is given as a probability, whereas the exploitability is given as a number between 0 (cannot be exploited) and 1 (fully exploitable). The concrete measures are shown in Table 3.

| Vulnerability | Likelihood of existence | Exploitability |
|---|---|---|
| CWE-20 | [0, 0.02] | 0.3 |
| CWE-89 | [0, 0.01] | 0.4 |

**Table 3 – Test measures**

### 3.3.4 Aggregating Test Measures

For the risk analysts the objective is to use the test results to reduce the uncertainty of the risk estimates. In our particular example, the objective is to reduce the probability interval [0.001, 0.1] of the leads-to relation between the two threat scenarios in Figure 13. This means that we aim to use the test measures as input for the estimation of this interval.

As mentioned above, this kind of aggregation from test measures to likelihood estimates, possibly via risk measures, may often have to be conducted manually and by expert judgments. In the example we illustrate the aggregation by a function that simply multiplies the test measures for each vulnerability and then adds up the results to deduce the eventual probability interval. Referring to the test measure values in Table 3 we get the following.

$$([0, 0.02] \cdot 0.3) + ([0, 0.01] \cdot 0.4) = [0, 0.006] + [0, 0.004] = [0, 0.01].$$

As explained in Section 3.2.2 we need to use also the likelihood estimate from the risk assessment as input when making any updates to the risk model. As shown in Figure 13 the original estimate was the interval [0.001, 0.1]. Combining the results we get the updated interval [0.001, 0.01]. The updated threat diagram, where we have also recalculated the likelihood estimates that follow the tested leads-to relation, is depicted in Figure 15.

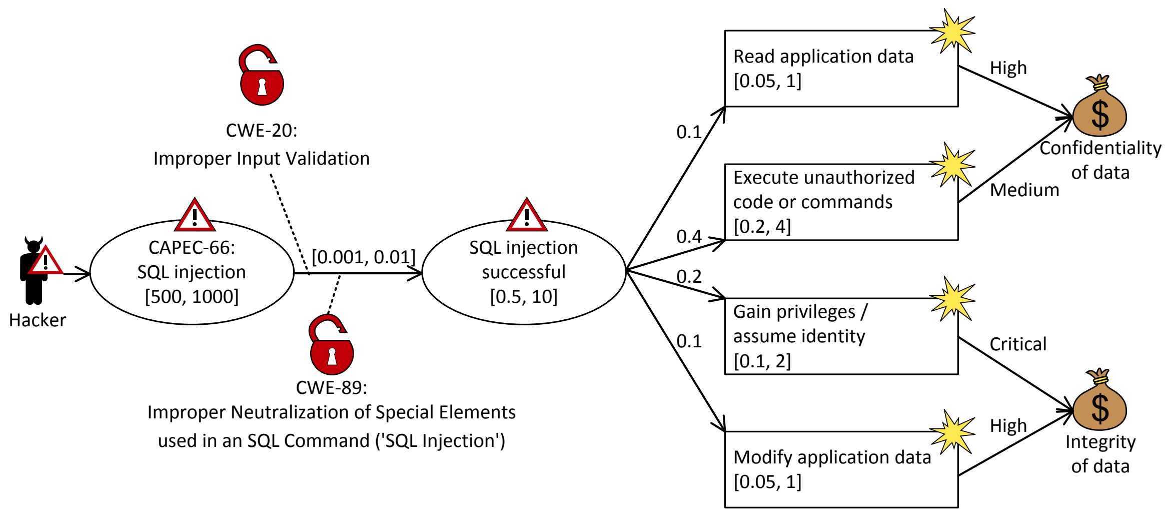


**Figure 15 – Threat diagram updated after testing**

The resulting and updated risk evaluation is shown in Figure 16. We see that the uncertainty of the estimates has been reduced to the extent that none of the identified risks span over all three risk levels. The risks *Execute unauthorized code or commands*, *Read application data* and *Modify application data* are not unacceptable, but *low* or *medium*.

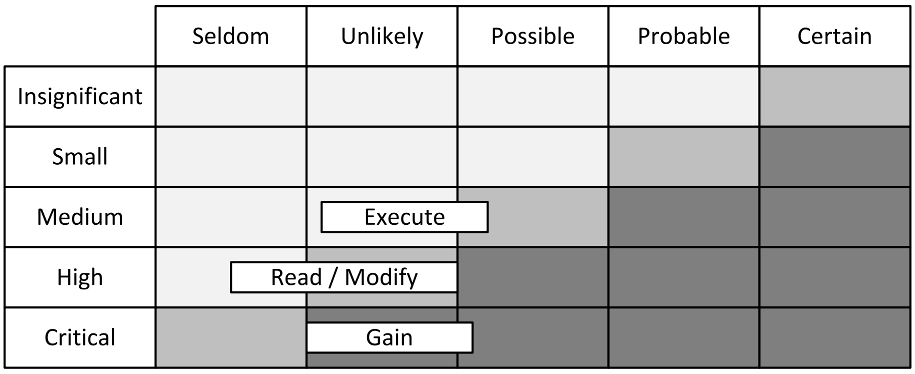


**Figure 16 – Updated risk evaluation**

If the updated estimates and the updated evaluation provides sufficient basis for the decision makers to determine for each risk whether it needs treatment, the risk assessment process can proceed to the treatment phase. Otherwise the analysts need to gather more empirical data, for example by more extensive security testing.

## 3.4 Conclusion

Test-based security risk assessment involves the systematic use of security testing for gathering data to support the risk estimation. Following the RASEN method, the test procedures are identified based on the risk models, and the security test results are in turn fed back to the risk model. This process requires that the test procedures are linked to the elements of the risk model under test and that we have data formats that can support the information exchange between the risk assessment and the testing.

In the RASEN method we use measures and metrics for representing security test results and information that is underlying the risk models and risk estimates. In this section we have explained and illustrated how the test measures and metrics are derived from the test procedures that in turn are derived from the security risk models. We have moreover explained how the risk analysts make use of the test measures and metrics to aggregate them and derive risk measures and metrics that can be used for the security risk assessment.

The RASEN techniques for test-based security risk assessment are being applied in the RASEN case studies, where we make use of test and risk measures. Currently the aggregation of test measures is not automated in the case studies, but rather conducted as expert judgments. The project aim to develop some automated support, but for this to be viable we need some formats for aggregation that can be easily reused in different risk assessments. Otherwise the specification and implementation of aggregation functions would be costly one-off tasks in each new security risk assessment.

# 4 Automated Risk Assessment with Testing

Especially for large scale systems, security risk assessment and security testing might be both difficult and expensive. Reusing already created artefacts in combination with automation might help to minimize costs. It also might help to reduce the dependency on the expertise, skills and accuracy of the analysts. Hence, it might help to reduce human errors in the entire assessment and testing process.

We consider security testing itself to be one possible way to make risk assessment more objective and more precise. There are other concepts and technologies which could be applied for the same purpose, including but not limited to formal verification and simulation. While formal verification might be hard or infeasible for complex large scale systems and while testing these systems might be expensive, simulations can help to deal with exactly those large systems and to overcome scaling problems.

It makes eventually sense to combine simulation and testing technologies in order to refine the risk assessment of systems which cannot be entirely tested. Combining simulation and security testing might lead to new concepts for their seamless integration into the risk assessment process. These are the basic ideas that inspired the development of the RACOMAT method.

## 4.1 The General RACOMAT Method

RACOMAT is an acronym for Risk Assessment COMbined with Automated Testing. The RACOMAT method integrates security testing tightly into incident simulations of a low level compositional security risk assessment. The method tries to automatically test exactly the most critical parts with reasonable effort and to improve the risk picture with objective test results. Figure 17 shows the entire process of the RACOMAT method.

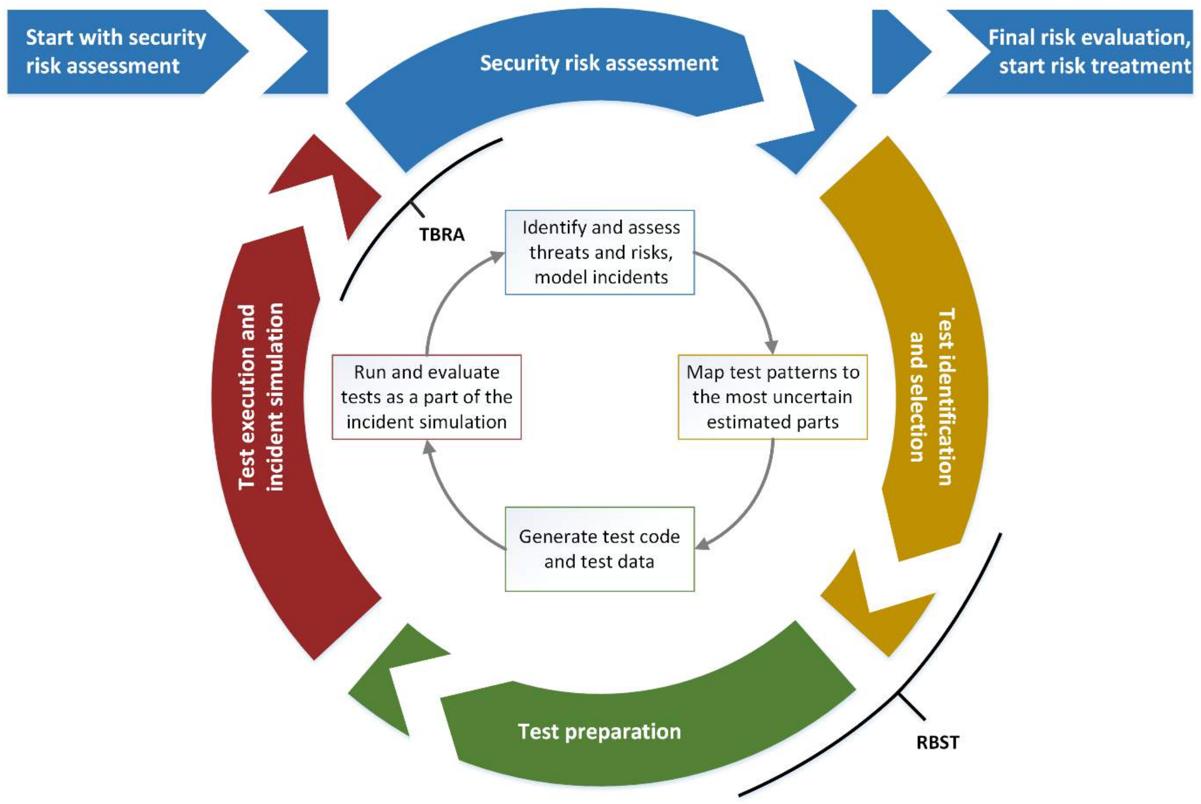

Start with security risk assessment
Security risk assessment
Final risk evaluation, start risk treatment
TBRA
Identify and assess threats and risks, model incidents
Map test patterns to the most uncertain estimated parts
Generate test code and test data
Run and evaluate tests as a part of the incident simulation
Test execution and incident simulation
Test identification and selection
Test preparation
RBST

**Figure 17 – The general RACOMAT method**

In spite of relying upon a specific risk assessment method, the general RACOMAT method can use different kinds of risk assessment methods, including fault tree analysis (FTA)[1], event tree analysis

(ETA) [2] and the CORAS method [7]. However, it requires risk assessment to support component based risk analysis and compositionality as described, for example, for the FTA extension in [9] or for the CORAS extension in [24]. Especially it must be possible to model dependencies between events (faults, incidents) precisely.

For automation, the risk assessment must be made on a low level. RACOMAT allows analysts to model close relations to parts and components of the systems that are analyzed. Therefore, RACOMAT introduces the concept of interfaces and ports which can be used to model the relations between risk analysis artefacts and the system.

Since low level risk assessment traditionally means a lot of effort, reusability of existing artefacts is a vital part of the RACOMAT method. The component based approach with risk composition in combination with libraries of existing risk analysis artefacts like attack patterns and catalogues of security test patterns should help to limit the effort. For some common types of components, the RACOMAT method suggests using predefined complete modules as defined in [5], for example.

The initial assessment has to be performed until it results in an event graph with dependencies between the events and likelihood notations for the occurrence of independent incidents.

Such a graph can be used to simulate the behavior of the system. Running Monte Carlo simulations as described in [25], likelihoods for dependent events can be calculated. The modeled dependencies and the likelihood estimates thereby simulate the actual system.

The idea to improve the risk picture is now to replace some parts of the simulation with testing the real system components. That is, instead of simulating whether some event occurred based upon random values and likelihood functions, the RACOMAT method tries to actually trigger the incident. Automated or at least semi-automated testing is done with the help of test patterns as described in RASEN deliverable D4.2.2 [17]. Eventually, it might be necessary to generate some base incidents during the tests. Therefore, the RACOMAT method uses the concept of testing stubs. These stubs are small programs that create the required incidents juts for testing.

As test results the RACOMAT method yields which incidents have occurred. Hence, it is possible to directly overtake the occurrence states of those incidents that are already modeled in the event graph into the incident simulation.

If likelihoods are expressed as functions like it is suggested in [25], it is possible to use the test results to interpolate a likelihood function that accurately replaces the tested component in future simulations. That is exactly how to update the event graph. Since there is now a new test-based interpolated likelihood function, it is possible to use this function within following simulation runs to emulate the behavior of the already tested part accurately.

The RACOMAT method continues by replacing the likelihood estimates for the next most uncertain asserted component with testing the corresponding real component in the next updated incident simulation.

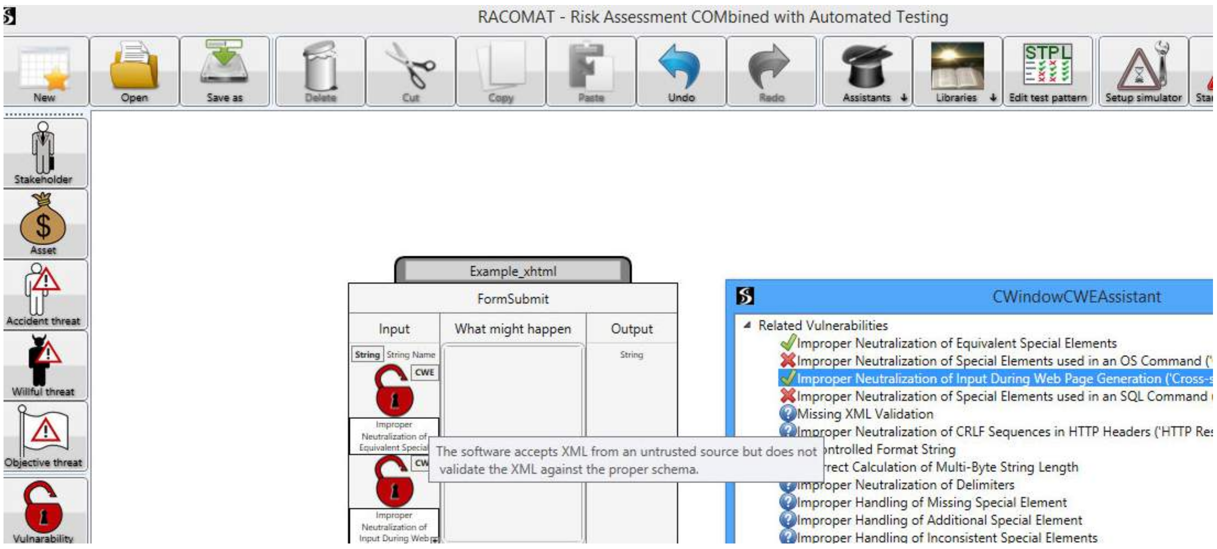
If all components have been tested or if the testing budget is used up, then the latest risk picture becomes the final test-based risk assessment result. Further risk management might continue with additional evaluation of the results and with risk treatment. However, this is beyond the scope of the RACOMAT methodology.

## 4.2 Specific Techniques Introduced with the RACOMAT Tool

Implementing one specific instance of the general RACOMAT method, the RACOMAT tool combines the general method with advanced techniques for a high level of automation. The RACOMAT tool uses fault tree analysis or CORAS with extensions for compositional risk assessment.

In order to reduce the manual effort of low level risk assessment, the RACOMAT tool integrates techniques for analyzing components automatically. Given (X)HTML pages, source code or compiled programs, it tries to identify the public interfaces of any components and especially the functions as well as ports that could be used for interaction with other components or users. An initial system model is generated without requiring manual actions.

The RACOMAT tool assists the risk analysts by suggesting all relevant faults or vulnerabilities, attack patterns and unwanted incidents for the identified system elements. The assistants shown in Figure 18 and Figure 19 take advantage of existing risk related libraries (e.g. MITRE CAPEC [12] and CWE [10] or BSI IT-Grundschutz [5]). The task of the analysts is not to find the relevant risk artefacts. It is rather to exclude the non-relevant artefacts. This assisted "negative", excluding risk assessment technique shown in Figure 18 is somehow similar to check lists, limiting the chance that relevant aspects are simply overlooked. Risk artefacts are added with simple drag and drop. Thereby, they can be immediately linked with the elements of the automatically generated system models.

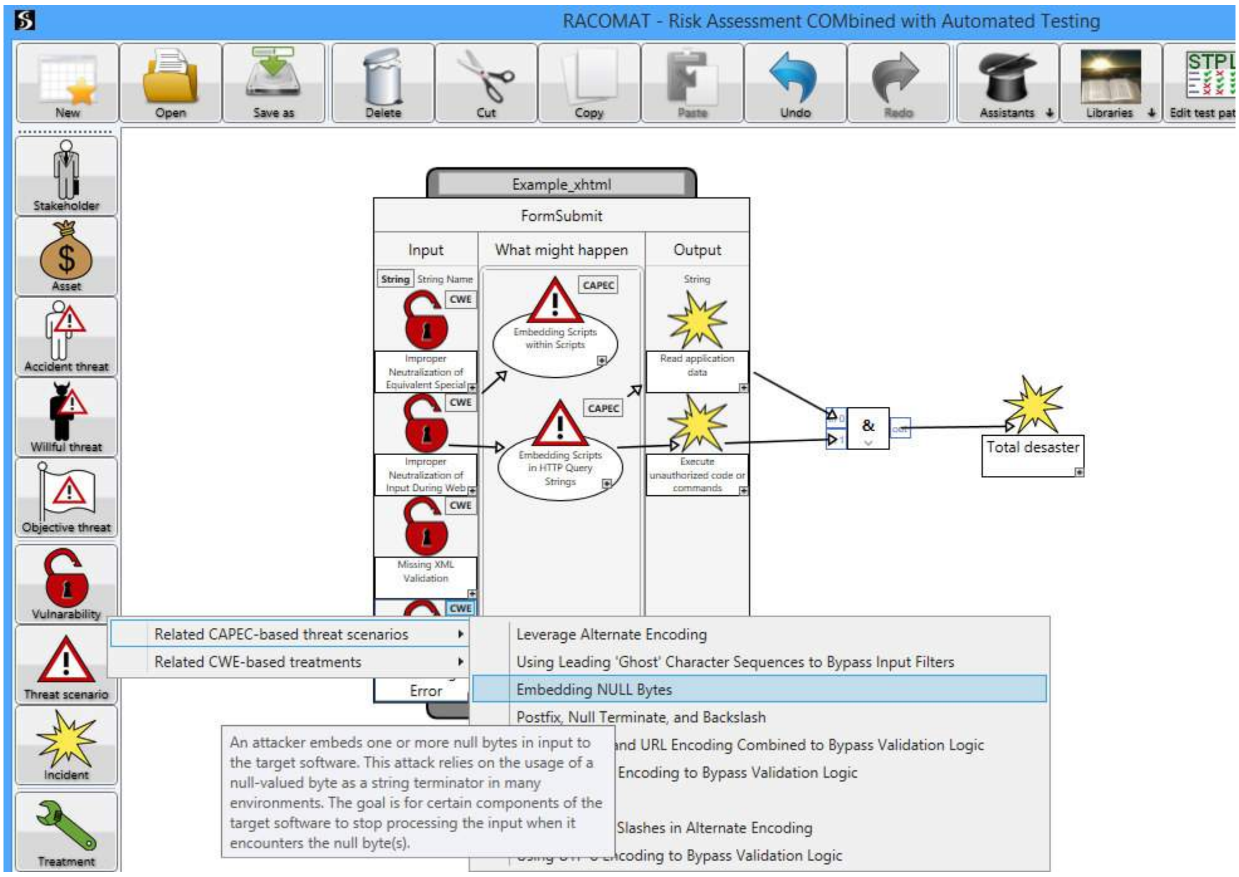


The software accepts XML from an untrusted source but does not validate the XML against the proper schema.

Related Vulnerabilities
- ✓ Improper Neutralization of Equivalent Special Elements
- ✗ Improper Neutralization of Special Elements used in an OS Command ('
- ✓ Improper Neutralization of Input During Web Page Generation ('Cross-s
- ✗ Improper Neutralization of Special Elements used in an SQL Command
- ? Missing XML Validation
- ? Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Res
- ntrolled Format String
- rect Calculation of Multi-Byte String Length
- ? Improper Neutralization of Delimiters
- ? Improper Handling of Missing Special Element
- ? Improper Handling of Additional Special Element
- ? Improper Handling of Inconsistent Special Elements

**Figure 18 – RACOMAT assistant suggesting relevant vulnerabilities for string input field in website**

Dependencies between faults or incidents can be modeled in detail using directed weighted relations and gates like shown in Figure 19. As required by the general RACOMAT method, the RACOMAT tool supports compositional risk analysis and it calculates likelihoods for dependent incidents using Monte Carlo simulations.


RACOMAT - Risk Assessment COMbined with Automated Testing
New
Open
Save as
Delete
Cut
Copy
Paste
Undo
Redo
Assistants
Libraries
Edit test pat
Stakeholder
Asset
Accident threat
Willful threat
Objective threat
Vulnerability
Threat scenario
Incident
Treatment
Example_xhtml
FormSubmit
Input
What might happen
Output
String
String Name
CWE
CAPEC
Embedding Scripts within Scripts
Read application data
Improper Neutralization of Equivalent Special
Improper Neutralization of Input During Web
Embedding Scripts in HTTP Query Strings
Execute unauthorized code or commands
Missing XML Validation
&
Total desaster
Error
Related CAPEC-based threat scenarios
Related CWE-based treatments
Leverage Alternate Encoding
Using Leading 'Ghost' Character Sequences to Bypass Input Filters
Embedding NULL Bytes
Postfix, Null Terminate, and Backslash
nd URL Encoding Combined to Bypass Validation Logic
Encoding to Bypass Validation Logic
Slashes in Alternate Encoding
coding to Bypass Validation Logic

An attacker embeds one or more null bytes in input to the target software. This attack relies on the usage of a null-valued byte as a string terminator in many environments. The goal is for certain components of the target software to stop processing the input when it encounters the null byte(s).

**Figure 19 – RACOMAT assistant suggesting threat scenarios related to some vulnerability**

Security test patterns are automatically associated with risk analysis artefacts as well as system model components (e.g. input and output ports) and their priority is calculated. If no appropriate test patterns exist in the library, the tool allows its users to create new test patterns within the tool and to upload them to the library for sharing. Given an appropriate test pattern, the test generation, execution and result aggregation are at least semi-automated. Support for creating and integrating test stubs that can generate base incidents for testing purpose is currently under development.

Test results can be integrated seamlessly into Monte Carlo simulations for calculating likelihood values. It is possible to update the risk graphs automatically with more precise likelihood estimates interpolated from test results or with new faults based on unexpected test results. Additional security testing metrics can be created and applied manually.

The RACOMAT tool and a tutorial video showing the tool in action are included in RASEN deliverable D3.3.2.

## 4.3 Interaction with Other Tools

The RACOMAT tool can be used as a stand-alone tool. It covers the entire process of combined test-based risk assessment (TBRA) and risk-based security testing (RBST) shown in Figure 17. Nevertheless, it is also possible to use other possibly more specialized tools for some steps in that process. In particular, the RACOMAT tool can be used in conjunction with the other tools developed and used within the RASEN project. Since the RACOMAT tool supports the entire process, it makes sense to use the RACOMAT tool as the central platform for the data exchange and for any other interaction between the tools. Figure 20 illustrates how such a risk assessment and security testing process using different tools and RACOMAT tool as central platform could work.

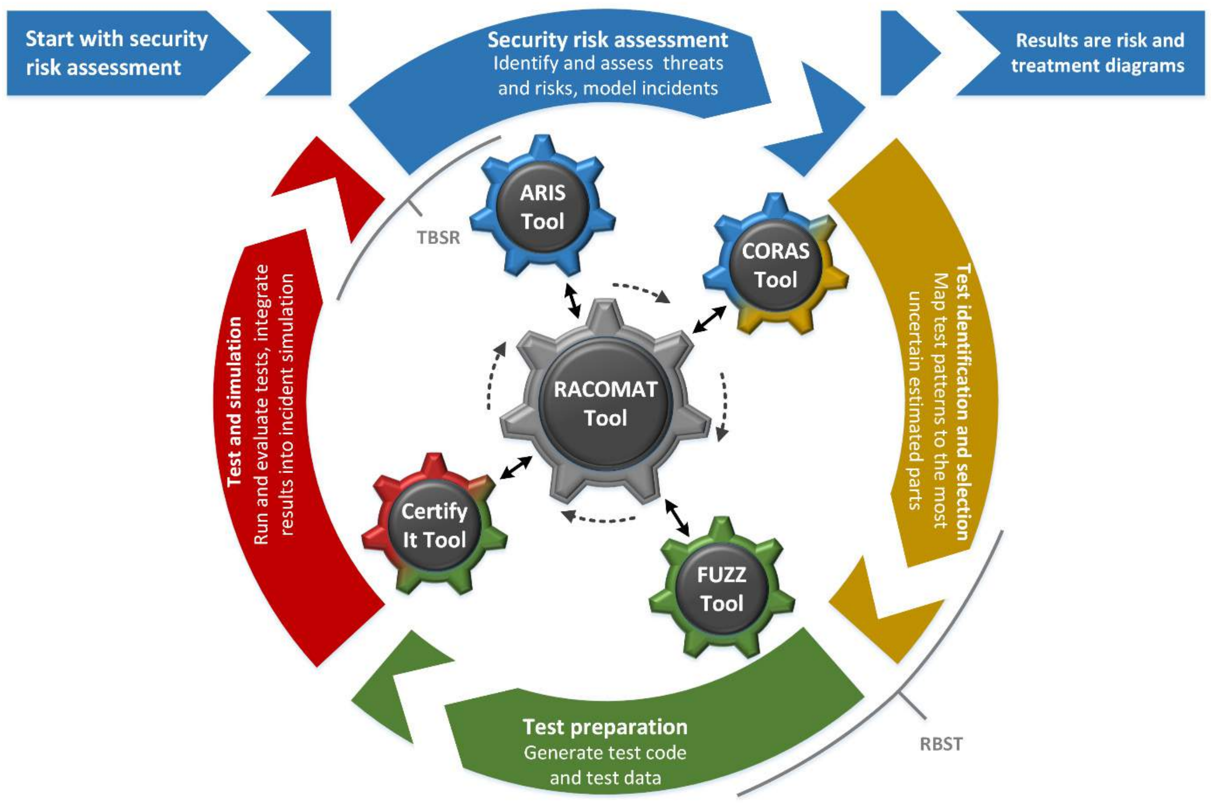
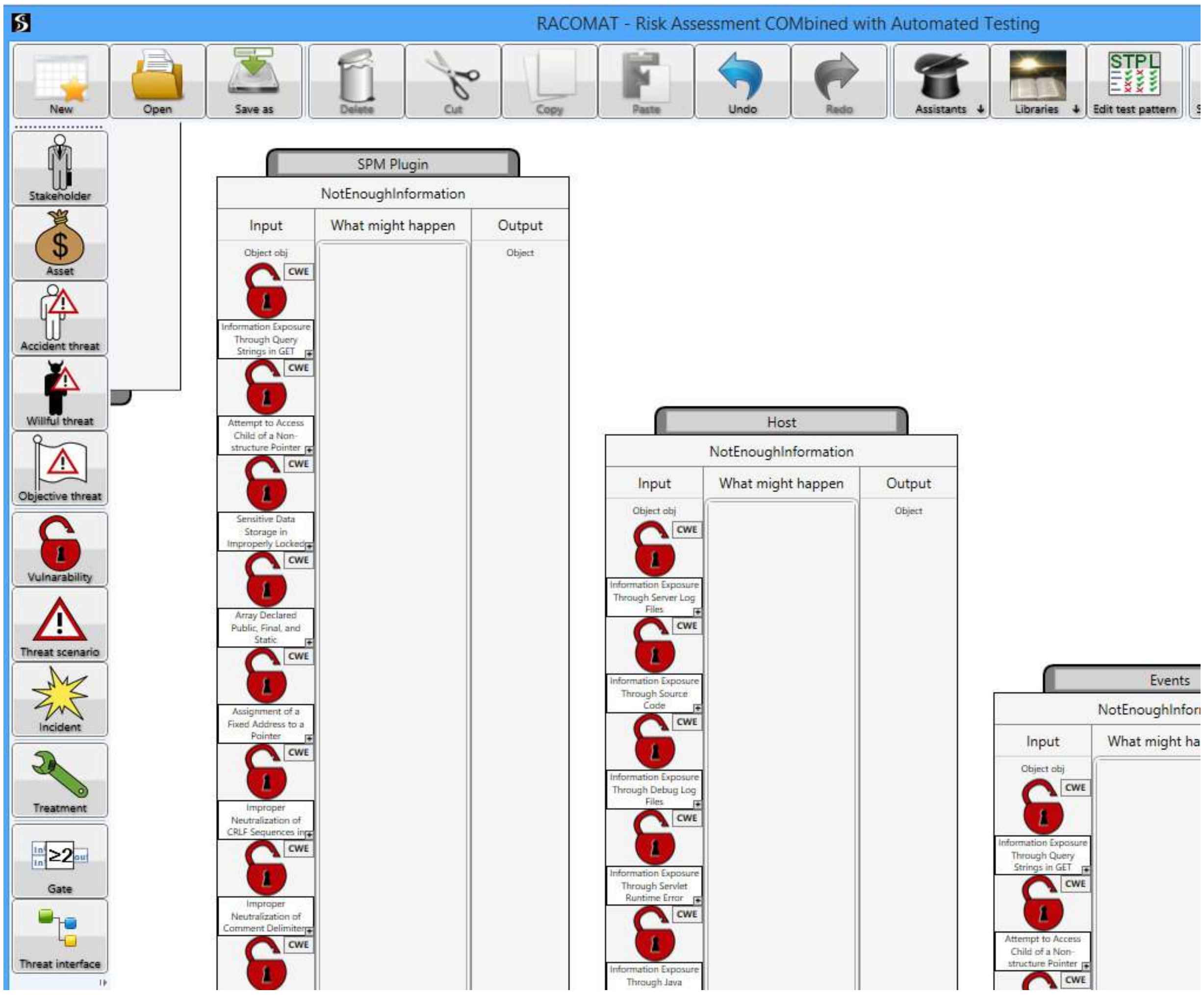

Start with security risk assessment
Security risk assessment
Identify and assess threats and risks, model incidents
Results are risk and treatment diagrams
TBSR
ARIS Tool
CORAS Tool
RACOMAT Tool
Certify It Tool
FUZZ Tool
Test identification and selection
Map test patterns to the most uncertain estimated parts
Test preparation
Generate test code and test data
RBST
Test and simulation
Run and evaluate tests, integrate results into incident simulation

**Figure 20 – The process with various tools**

For instance, the RACOMAT tool already supports the JSON (JavaScript Object Notation) format of the ARIS tool for data exchange. Figure 21 shows results of import from ARIS to RACOMAT. Support for data exchange with the CORAS tool is currently being developed and integrated into the RACOMAT tool. The RACOMAT tool is also going to provide tracing features to keep track of elements that are exported and later imported again, eventually altered by some external tool.



RACOMAT - Risk Assessment COMbined with Automated Testing
New
Open
Save as
Delete
Cut
Copy
Paste
Undo
Redo
Assistants
Libraries
Edit test pattern
Stakeholder
Asset
Accident threat
Willful threat
Objective threat
Vulnerability
Threat scenario
Incident
Treatment
Gate
Threat interface
SPM Plugin
NotEnoughInformation
Input
What might happen
Output
Object obj
Object
CWE
Information Exposure Through Query Strings in GET
Attempt to Access Child of a Non-structure Pointer
Sensitive Data Storage in Improperly Locked
Array Declared Public, Final, and Static
Assignment of a Fixed Address to a Pointer
Improper Neutralization of CRLF Sequences in
Improper Neutralization of Comment Delimiter
Host
NotEnoughInformation
Input
What might happen
Output
Object obj
Object
Information Exposure Through Server Log Files
Information Exposure Through Source Code
Information Exposure Through Debug Log Files
Information Exposure Through Servlet Runtime Error
Information Exposure Through Java
Events
NotEnoughInfor
Input
What might ha
Object obj
Information Exposure Through Query Strings in GET
Attempt to Access Child of a Non-structure Pointer

**Figure 21 – Initial threat interfaces imported from ARIS**

## 4.4 Conclusion

Using security testing to improve fault and security risk incident simulations, the RACOMAT method introduces an intuitive way for how risk assessment and security testing can interact with one another. The RACOMAT tool shows how this method can be implemented and how the entire process can be automated to a great extent. However, automation requires adequate security test patterns and testing metrics. Currently there are only few existing, but with the help of the RACOMAT tool it is now possible to quickly develop new ones.

Bearing in mind the ongoing work to provide interoperability with other tools which are more specialized on certain steps, the RACOMAT tool might in the future become a central platform capable to manage entire combined risk assessment and security testing processes using multiple heterogeneous tools.

# 5 Risk Model Encapsulation and Composition

The criticality of risk management is evident when considering the information society of today. Information systems become ever more complex, heterogeneous, dynamic and interoperable. Businesses, enterprises, governments and many other stakeholders rely more and more on the availability of services and information over the Internet, with Cloud services as a prominent example. Managing risks in such a setting is extremely challenging, and established methods and techniques are often inadequate. The main problems are that the overall risk picture becomes too complex to understand, and that the risk picture quickly and continuously changes and evolves.

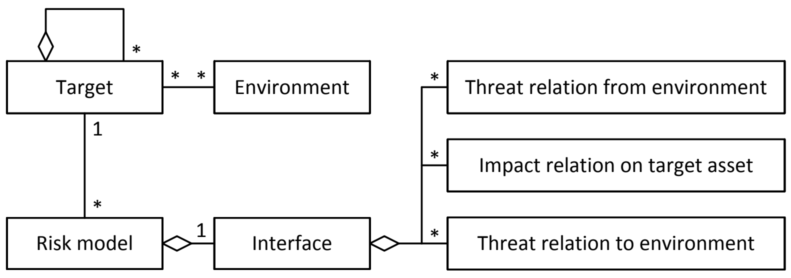
The challenge we address in this section is how to facilitate a compositional [21] approach to risk assessment by applying the principle of *encapsulation*. Following a divide-and-conquer strategy we aim for an approach to risk assessment where separate parts of a system can be analyzed individually. Compositional techniques should then enable a systematic and methodic composition of the individual risk models in order to derive the overall combined result without having to reconsider the details of the individual models.

While compositionality is an established technique in system development, there is very little support for this within risk management and assessment [20]. The method and techniques presented in this section builds on the formal foundation for compositional risk assessment presented in RASEN deliverable D3.2.1. For further details and a more elaborate presentation of the application of the techniques, the reader is referred to the publication on which this section is based [20].

## 5.1 Risk Model Encapsulation

The objective of introducing the notion of risk model encapsulation is to allow different risk models to be composed without having to know or understand all the interior details of the individual models. For this purpose we need a notion of risk model interface that contains all and only the information that is needed for the risk model composition. A further challenge that needs to be tackled is how to take into account possible dependencies between the individual risk models. Each sub-target is analyzed separately, but the other sub-targets belong to the environment of the sub-target being analyzed. This means that the other sub-targets can serve as environmental causes of risks that must be taken into account, and that the sub-target being analyzed can be the cause of risks for the sub-targets in its environment.

In the UML [13] class diagram of Figure 22, the term *target* denotes the target of analysis. The goal of the risk assessment is to build the *risk model* for the target. The target may be decomposed onto a number of smaller parts (that we often refer to as sub-targets). There are two crucial features of our approach to risk model encapsulation. First, for each target we need to understand how it relates to its *environment*. Second, we need a precise notion of an *interface* which consists of the risk information that is needed in order to compose the risk model in question with other risk models.



Target
Environment
Threat relation from environment
Impact relation on target asset
Risk model
Interface
Threat relation to environment
*
1

**Figure 22 – Risk model interface**

The interface consists of three sets of ingredients. The *threat relation from the environment* represents ways in which the environment may influence the risk model of the target. The *impact relation on target asset* represents potential harm on something of value inside the target. The *threat relation to environment* represents ways in which the target may influence the risk model of the environment.

## 5.2 Target Decomposition

Our method for compositional risk assessment follows the principles and guidelines defined by established standards such as ISO 31000 [3] and ISO/IEC 27005 [4]. For the details on our method, the reader is referred to [20]. In the rest of the section we demonstrate it by showing an example.

As a first step the overall target of analysis must be specified along with the assets to be protected. Our example is from the petroleum industry. Accidents on oil and gas rigs can have large consequences in terms of loss of life, damage to the environment and economic loss. Non-routine work that takes place on a rig, such as welding or replacement of defect gas detectors, may increase the risk. All such work therefore requires a work permit (WP). Every 12th hour a WP meeting is held to decide which work permits to release for the next shift.

In the following we assume that a petroleum operator has initiated a project in collaboration with a software tool and service provider to update their ICT system for WP management. In addition to functionality for registering, releasing and rejecting WP applications, the system will provide decision support in the form of an automated smart agent that collects relevant information for each WP application and provides advice to the human decision makers.

The collaboration diagram to the left of Figure 23 gives an overall view of the system. The class **RigSystem** represents all ICT infrastructure related to WPs that are installed on the rig itself. **WPAgent** represents the automated agent, which will be developed and maintained by the software provider represented by the **WPAgent maintainer**. **WeatherService** is an internet-based meteorological service offering weather forecasts.

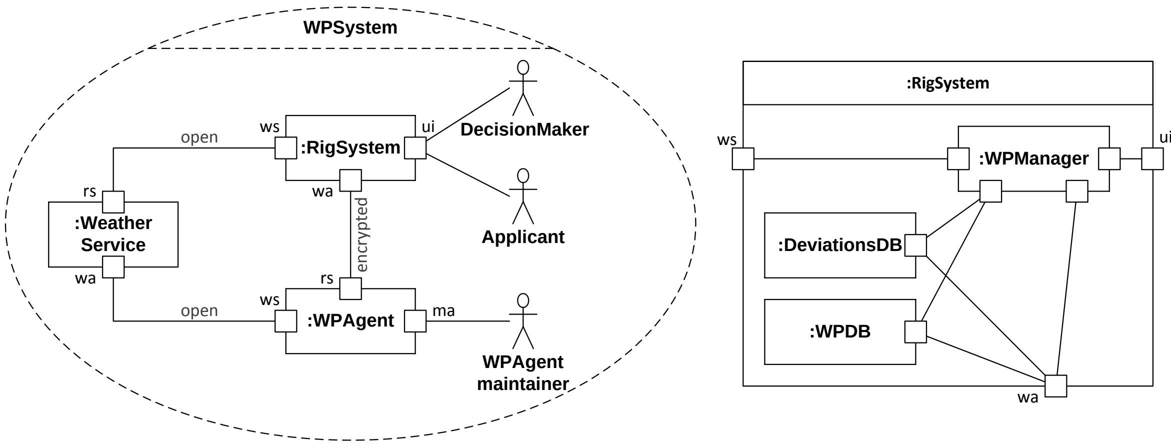


**Figure 23 – Structure of the WP system**

The internal details of **RigSystem** are shown to the right in Figure 23. Each of the internal components is available to the WP agent via the **wa** port. **WPManager** handles WP applications and release/reject decisions. **DeviationsDB** is a database where deviations related to the state, maintenance, testing, etc. of equipment on the rig are recorded. **WPDB** is a database that stores all WPs and related information. For further details about the target of analysis, including detailed sequence diagrams specifying the application process, the reader is referred to [20].

The next step is the identification of the assets for the overall target of analysis. There are of course a number of critical information and service assets in the WP scenario. For the purpose of the example we select only a few that we focus on. Considering the rig system it is obvious that availability of the WP data and availability of the WP advice are essential for both WP manager and for the decision maker. The availability of WP data is also essential for the WP agent that needs data for creating the advice. Considering the WP system as a whole, it is also critical to ensure the dependability of the WP agent. Because the WP agent is a software for automated decision support, the integrity of the software—including the implemented algorithms—needs to be protected. In the WP system analysis we are concerned about information security risks with respect to these assets.

The target decomposition is guided by both the identified assets and the system architecture. In Figure 24 we have specified the assets and placed them on the part of the target to which they belong. *Availability of WP data* concerns the information that is stored by the WPDB, whereas *availability of WP advice* concerns the result that is presented to the decision maker by the WP manager. The *dependability of WPAgent* and the *integrity of WPAgent software* belong to the WP agent.

Based on the identified assets we have decomposed the target into two components as indicated in Figure 24. Two of the assets are associated with the rig system and two of them with the WP agent and its communication line to the rig system. In the remainder of the section we refer to the former as *Component A* and to the latter as Component B. Note that in this analysis the Internet weather service is part of the environment of the overall target of analysis.

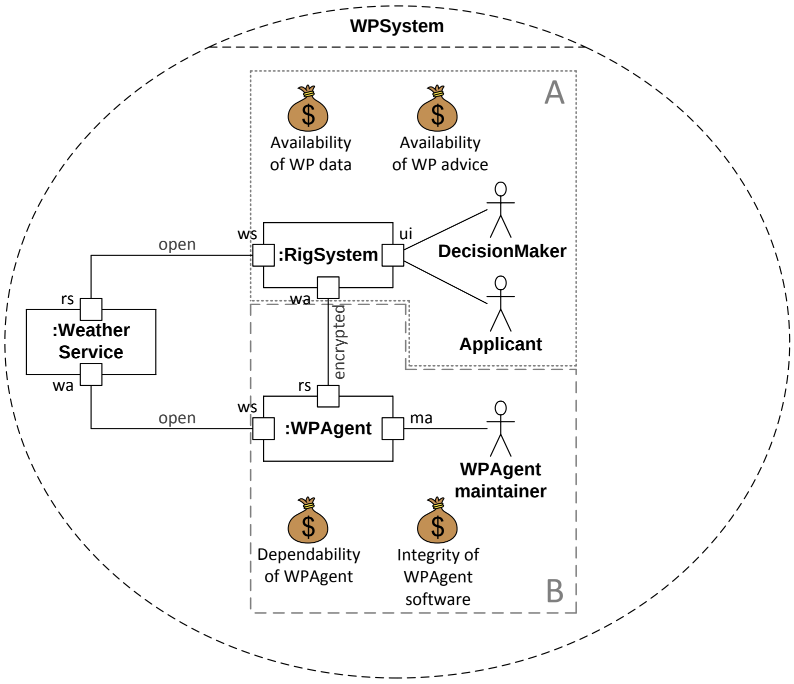


**Figure 24 – Target decomposition**

## 5.3 Sub-Target Risk Modeling

In the following we give a stepwise introduction to how we do the risk assessment for individual sub-targets by describing three different cases. First, the simple situation where all threats and assets are internal, i.e. risk identification with respect to threats and assets only within the sub-target. Second, we consider also external threats, i.e. causes that may stem from other sub-targets or from the environment of the overall target of analysis. Third, we address the general situation where we consider also environment assets, namely the assets of other sub-targets for which the sub-target in question can act as a source of risk. Note that this stepwise introduction is for presentation purposes and does not indicate a specific methodic order.

### 5.3.1 Internal Threats and Assets

Figure 25 shows our format for compositional risk modeling. Note that although we have used CORAS [7] for the risk modeling, our principles and modeling format for risk model encapsulation and composition can be applied using also other notations for risk modeling. The format consists of three compartments, where the middle compartment includes all the threats, vulnerabilities, assets, etc. that are internal to the sub-target in question. The diagram in Figure 25 is for sub-target A as the naming at the top indicates. In the compartment to the left we model environment threats and in the compartment to the right we model environment assets. Neither of these is relevant when considering internal threats and assets only.

One of the identified risks for sub-target A is that *WP advice cannot be accessed from WPManager*, which could be due to a software error that leads to malfunctioning of the WP manager. After such risk identification and modeling, the risk assessment proceeds with the risk estimation. In the example we have used frequencies for likelihood estimation and a scale of the three levels high (H), medium (M) and low (L) for the consequences. In the example we have estimated that *WP data cannot be accessed from WPManager* occurs 16 times per year. The reader is referred to existing literature on CORAS [7][23] for the techniques and rules for risk estimation. The likelihood of the other incident, however, is not estimated at this point. This is because the analysts know that the availability of the WP advice depends also on the WP agent. We therefore need to take into account also environment threats.

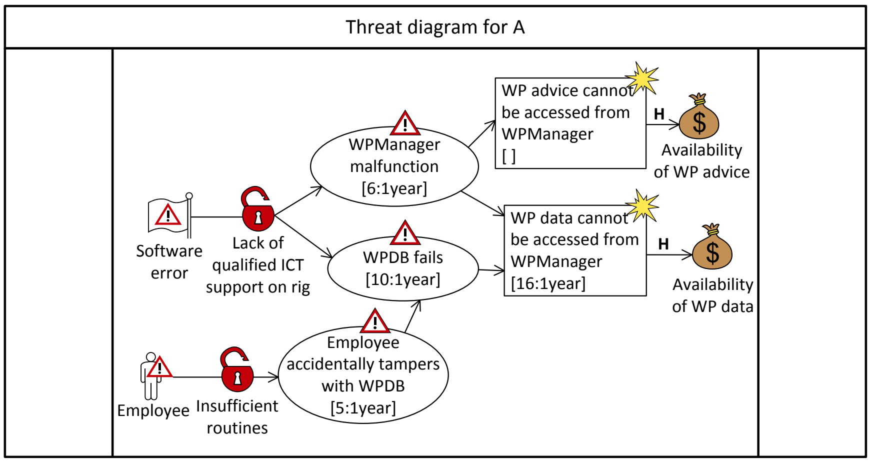

Threat diagram for A
WP advice cannot be accessed from WPManager [ ]
H
Availability of WP advice
WPManager malfunction [6:1year]
WP data cannot be accessed from WPManager [16:1year]
H
Availability of WP data
Software error
Lack of qualified ICT support on rig
WPDB fails [10:1year]
Employee accidentally tampers with WPDB [5:1year]
Employee
Insufficient routines

**Figure 25 – Internal threats and assets**

### 5.3.2 Environment Threats

Figure 26 exemplifies an environment threat with respect to sub-target A, namely that *WPAgent fails to deliver advice*. Importantly, because the threat occurs outside of A the estimation of its likelihood is not part of the risk assessment of A. Instead the variable $x_1$ is used such that we get a parameterized specification of the likelihoods of the scenarios and incidents that this threat may cause. For example, using the CORAS rules for risk calculation, the estimated frequency of the incident *WP advice cannot be accessed from WPManager* is $x_1 + 6$ occurrences per year.

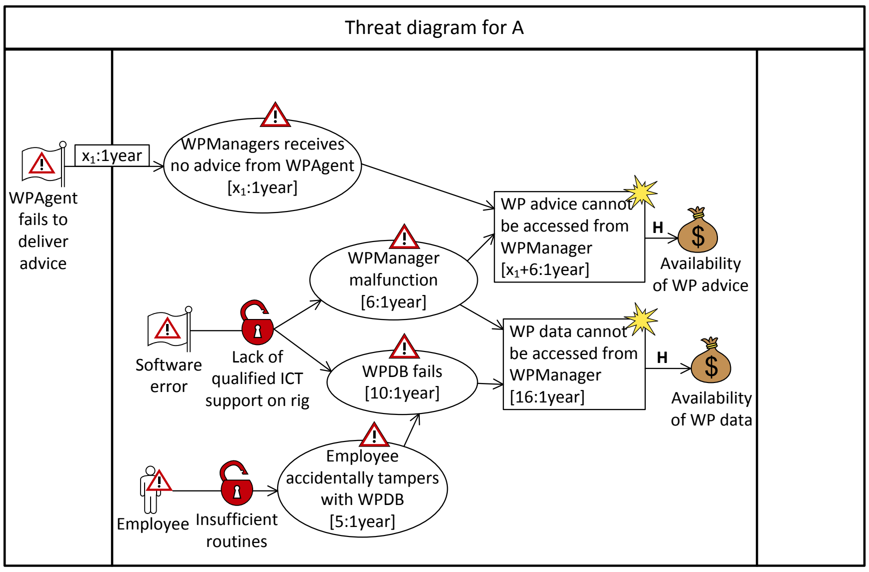

Threat diagram for A
WPAgent fails to deliver advice
x1:1year
WPManagers receives no advice from WPAgent [x1:1year]
WP advice cannot be accessed from WPManager [x1+6:1year]
H
Availability of WP advice
WPManager malfunction [6:1year]
WP data cannot be accessed from WPManager [16:1year]
H
Availability of WP data
Software error
Lack of qualified ICT support on rig
WPDB fails [10:1year]
Employee accidentally tampers with WPDB [5:1year]
Employee
Insufficient routines

**Figure 26 – Environment threats**

As we will show below the estimation of $x_1$ is done as part of the risk assessment of sub-target B, and this input is used when composing the threat diagrams to generate the risk picture for the overall target.

## 5.3.3 Environment Assets

In order to understand and analyze how one sub-target can act as an environment threat for another sub-target we need a systematic way to systematically consider all other sub-targets. Our approach to do this is to take into account all assets of the overall target in each individual risk assessment, while still distinguishing between the internal assets and the environment assets.

This is illustrated for sub-target A in the upper diagram of Figure 27. One of the assets that do not belong to A is *Dependability of WPAgent*. In the diagram this asset is placed in the compartment to the right. As part of the risk assessment of sub-target A we identify all incidents that may have an impact on any of the environment assets. In the example, one such incident is *Loss of WPDB*. Note importantly that the consequence estimation for the environment assets is not done as part of the risk assessment for the sub-target in question; exactly how incidents of the sub-target may impact assets of other sub-targets need to be analyzed as part of the risk assessment for each of the impacted sub-targets.

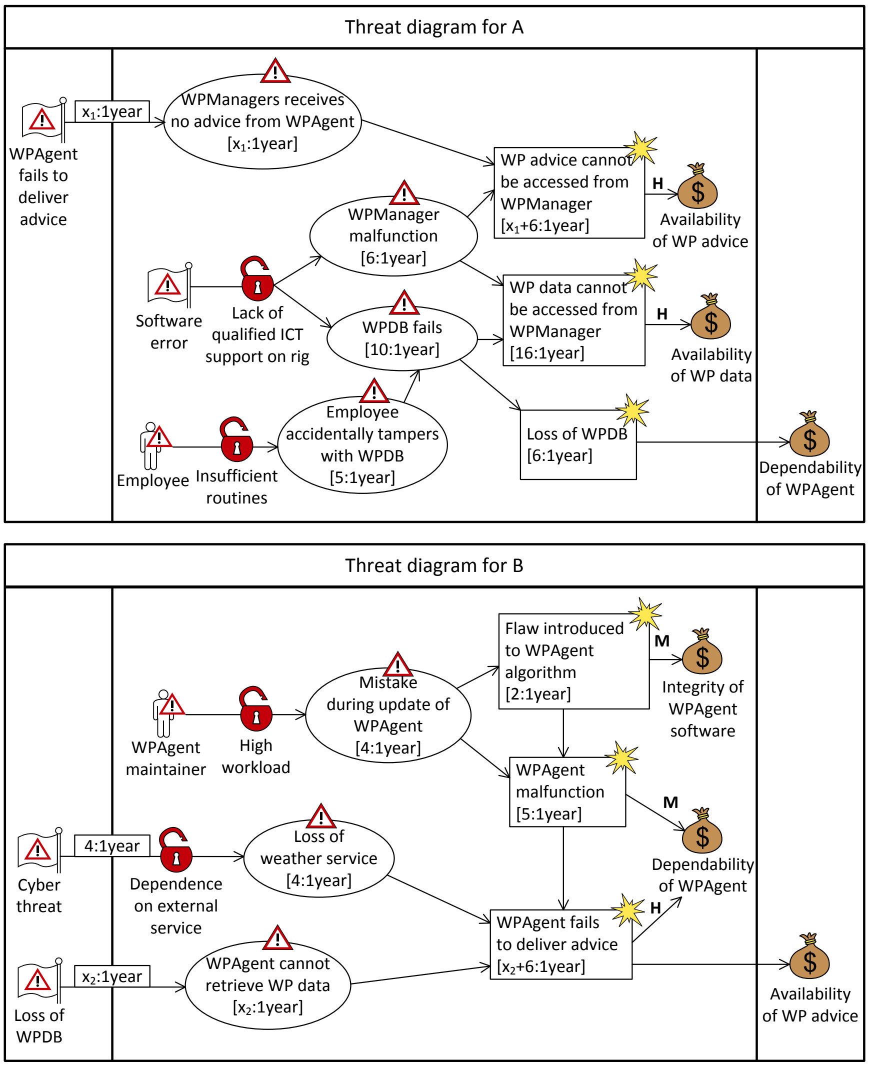

Threat diagram for A
WPAgent fails to deliver advice
x1:1year
WPManagers receives no advice from WPAgent [x1:1year]
WP advice cannot be accessed from WPManager [x1+6:1year]
H
Availability of WP advice
WPManager malfunction [6:1year]
Software error
Lack of qualified ICT support on rig
WPDB fails [10:1year]
WP data cannot be accessed from WPManager [16:1year]
H
Availability of WP data
Employee
Insufficient routines
Employee accidentally tampers with WPDB [5:1year]
Loss of WPDB [6:1year]
Dependability of WPAgent
Threat diagram for B
Flaw introduced to WPAgent algorithm [2:1year]
M
Integrity of WPAgent software
WPAgent maintainer
High workload
Mistake during update of WPAgent [4:1year]
WPAgent malfunction [5:1year]
M
Dependability of WPAgent
Cyber threat
4:1year
Dependence on external service
Loss of weather service [4:1year]
WPAgent fails to deliver advice [x2+6:1year]
H
Loss of WPDB
x2:1year
WPAgent cannot retrieve WP data [x2:1year]
Availability of WP advice

**Figure 27 – Environment assets**

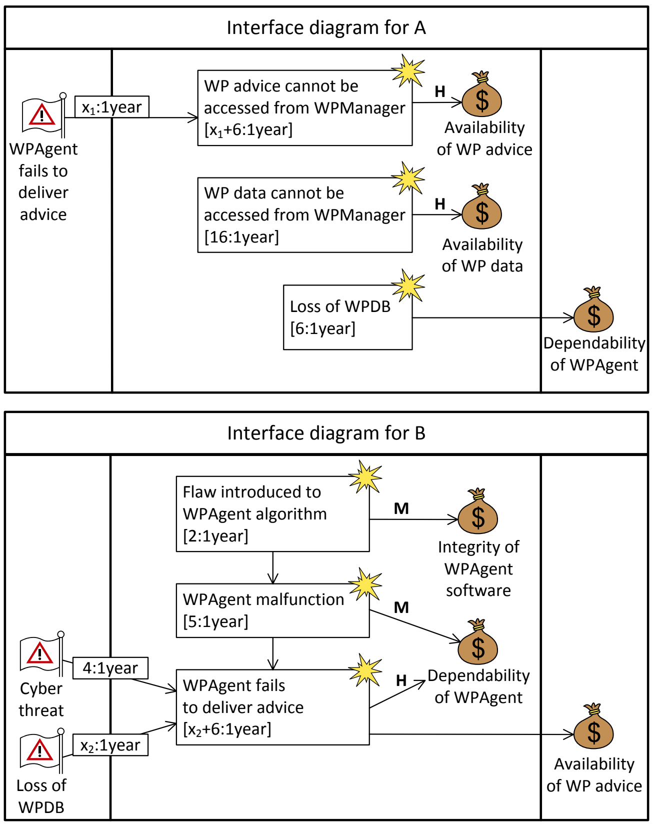
The lower diagram of Figure 27 exemplifies a completed threat diagram for sub-target B. We see here that the incident *WPAgent fails to deliver advice* may impact the environment asset *Availability of WP* *advice*. This asset belongs to A, which is why this incident occurs as an external threat in the threat diagram for sub-target A shown in the upper part of Figure 27. From the diagram for sub-target B we also see that incidents of one sub-target may impact its own assets as well as environment assets. We also see two environment threats, namely *Cyber threat* and *Loss of WPDB*. The latter stems from A, whereas the former stems from the environment of the overall target.

Whereas the likelihood estimation of external threats from sub-target A is done as part of the assessment of A, we may do a separate likelihood estimation for the threats from the environment of the overall target. This is exemplified for the cyber threat where the frequency is *4:1 year*.

## 5.4 Risk Composition

The threat diagrams introduced above give the white-box view of the risk model for each sub-target; their purpose is to support the full risk assessment of the sub-targets, including all the internal threats, vulnerabilities and threat scenarios. To facilitate their composition, however, we create their corresponding interface diagrams.

The interface diagrams for A and B are depicted in Figure 28. The interface diagrams contain the information that is needed to compose the different diagrams to yield the overall risk picture, and to document all of the risks with their risk levels. When composing the threat interface diagrams the variable $x_2$ in Figure 27 is instantiated with the value 6 from the incident *Loss of WPDB* for sub-target A. The likelihood of the unwanted incident *WPAgent fails to deliver advice* is then calculated by $x_2 + 6$, which gives *12:1 year*.



Interface diagram for A
$x_1$:1year
WPAgent fails to deliver advice
WP advice cannot be accessed from WPManager [$x_1$+6:1year]
H
Availability of WP advice
WP data cannot be accessed from WPManager [16:1year]
H
Availability of WP data
Loss of WPDB [6:1year]
Dependability of WPAgent
Interface diagram for B
Flaw introduced to WPAgent algorithm [2:1year]
M
Integrity of WPAgent software
WPAgent malfunction [5:1year]
M
Dependability of WPAgent
Cyber threat
4:1year
WPAgent fails to deliver advice [$x_2$+6:1year]
H
Loss of WPDB
$x_2$:1year
Availability of WP advice

**Figure 28 – Interface diagrams**

The resulting threat interface diagram for A and B composed, and hence for the overall target of analysis, is depicted in Figure 29. Since the diagram covers the whole target the set of environment assets is empty. Moreover, the only environment threat is the one that belongs to the environment of the overall target.

The interface diagram for the full target shows all unwanted incidents with respect to the assets we identified during the context establishment. It also shows the likelihood and consequence estimates for each of the incidents. Because a risk is defined as an unwanted incident with its likelihood and consequence, we have in our example identified five risks. The risk levels are calculated by using a risk function such as a risk matrix.

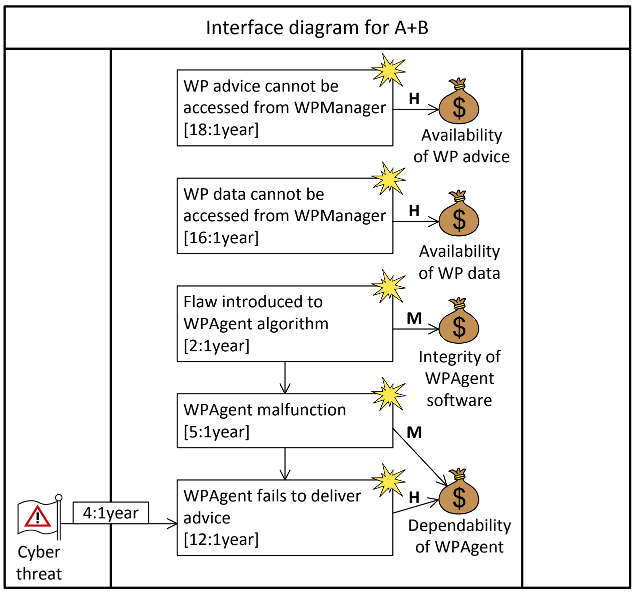

Interface diagram for A+B
WP advice cannot be accessed from WPManager [18:1year]
H
Availability of WP advice
WP data cannot be accessed from WPManager [16:1year]
H
Availability of WP data
Flaw introduced to WPAgent algorithm [2:1year]
M
Integrity of WPAgent software
WPAgent malfunction [5:1year]
M
WPAgent fails to deliver advice [12:1year]
H
Dependability of WPAgent
4:1year
Cyber threat

**Figure 29 – Interface diagram for the composed risk models**

## 5.5 Conclusion

In this section we have presented our first steps towards a method for compositional risk assessment. At the core of the approach is a novel notion of risk model encapsulation, where only the elements that are essential for composition are exposed through an explicitly defined risk model interface, while internal details are hidden. All one needs to know in order to compose risk models is the contents of their interfaces. By hiding the internal details we make it easier for practitioners to compose risk models, while at the same time reducing the size and complexity of the resulting model. An added benefit is that a risk model interface contains the information that would typically be of interest for managers and decision makers who often have little time and have not themselves taken part in the risk assessment.

Encapsulation is a key reason for the success of object-oriented programming. We believe that significant benefits can be achieved by introducing this concept into risk management and analysis. We are not aware of any other work where this has been attempted. For a presentation and discussion of related work we refer to [20]. Unlike our approach, none of the referred works provides a notion of encapsulation in the sense of exposing only the risk model elements needed for composition while hiding internal elements. Instead, they focus primarily on likelihood or risk level assessment in a component-based setting. While this is an important ingredient of component-based risk analysis, the lack of an encapsulation mechanism complicates composition; it means that composed models may become very large and complex, which in turn may lead to scalability problems.

# 6 Continuous Risk Assessment

Managing risk of systems that rapidly evolve or have environments that evolve is challenging since also the security risks may be evolving. This motivates the development of risk assessment techniques that facilitate a continuous assessment of systems where the risk picture can be updated without having to conduct a full risk assessment from scratch every time. There are various ways of handling system or environment changes and keeping the risk picture up-to-date. One approach that we continue to investigate in the RASEN project is the use of compositional risk assessment as presented in Section 5 and in deliverable D3.2.1 [18]; compositional techniques have the potential to facilitate continuous risk assessment by allowing only the parts or aspects of the system that have changed to be reassessed. Another approach is the use of traceability links between the documentation of the system and the risk model [8][22]. This facilitates efficient and systematic updating of the risk picture by the tracing of changes from the system to the relevant risks. Investigating the latter techniques is outside the scope of the RASEN project, although related to our objectives of facilitating continuous risk assessment. The third approach, which we also address in this project, is the use of key indicators that can be monitored. While monitoring the indicators the values can be used to update the existing risk picture at desired time intervals. This requires means for aggregating key indicator values to derive the values that can be fed to the risk models. As discussed in deliverable D3.2.1 this kind of indicator aggregation is similar to the aggregation of test measures presented in Section 3. The means for gathering the indicator values are of course different from the testing. The objective is also different since the security testing can be used to complement the assessment results, whereas the indicator monitoring is used to enable risk monitoring and continuous updates of the risk levels.

In Section 6.1 we give a brief overview of our approach to continuous risk assessment by indicator monitoring. In Section 6.2 through Section 6.4 we describe each step and the produced artefacts in more details. Finally we conclude in Section 6.5.

## 6.1 Method Overview

For risk monitoring to be possible there must be a monitoring infrastructure in place offering a palette of continuously monitored key indicators that can be selected from. The development of such an infrastructure is outside the scope of the RASEN project. We are instead concerned with how such indicators can be aggregated to update the risk model.

Before the risk monitoring can start we need to prepare by developing the risk model for the risks to be monitored, we need to identify the relevant key indicators, and we need to specify the functions for aggregating the indicator values. More specifically, the preparations consist of the following three steps.

1. **Initial risk assessment**. The purpose of this step is to identify and document the current risk picture so as to understand what is to be monitored and how it can be monitored. The initial risk assessment also includes the estimation of likelihoods and consequences in order to prioritize what to monitor, and also to provide data that can be used to validate the indicators and aggregation functions.
2. **Key indicator identification**. This is done based on the risk model and risk assessment results of the previous step. The indicators can be related to any of the risk model elements, such as threats, vulnerabilities, threat scenarios and unwanted incidents.
3. **Specify aggregation functions**. The purpose of this step is to define the mapping from the values of the monitored key indicators to the estimates that are fed to the risk model.

Note that while the aggregation of key indicator values in general can be used to derive any estimate for the risk model, including likelihoods, consequences and risk levels, the RASEN project focus on the aggregation of indicator values to likelihood values.

In the next subsections we give an example-driven presentation of the method and the produced model artefacts. The example is of a web-application for which the end-users need to have a user account and a username and password to access. Note that the risks and the estimates are made up for the example and not the real values from a real case.

## 6.2 Initial Risk Assessment

The initial risk assessment can be conducted using any available method and risk modeling language. In our example we have used the CORAS approach [7]. As the purpose here is not to present the risk assessment as such, we introduce only the results of the risk assessment that are used for the monitoring.

The initial risk assessment typically includes several assets that must be identified and documented. In our example we show only one asset, namely *integrity of data*. The risk assessment also includes the specification of scales for likelihoods and consequences, as well as the definition of the risk evaluation criteria. For the purposes of this section we only need the likelihood scale. The scale is of the five levels *seldom*, *unlikely*, *possible*, *probable* and *certain*. For the precise definition of each value we refer to Table 2 in Section 3.

The risk identification involves the identification and modeling of threats, vulnerabilities, threat scenarios and unwanted incidents. The result is exemplified with the CORAS threat diagram in Figure 30. The diagram shows two kinds of attacks on the web-application. On the one hand, a hacker may conduct an SQL injection attack which may lead to the risk of unauthorized modification of application data. On the other hand, a hacker or other intruder may gain unauthorized access by taking over a user account, either by a brute force attack on the user credentials or by someone accessing an active session on a shared workstation. The latter may happen when a user fails to log off an active session before leaving the workstation. The diagram also shows the likelihood estimates. The estimates for the threat scenarios and unwanted incidents are in terms of frequencies as defined in Table 2, whereas the conditional likelihoods on the leads-to relations are in terms of probabilities.

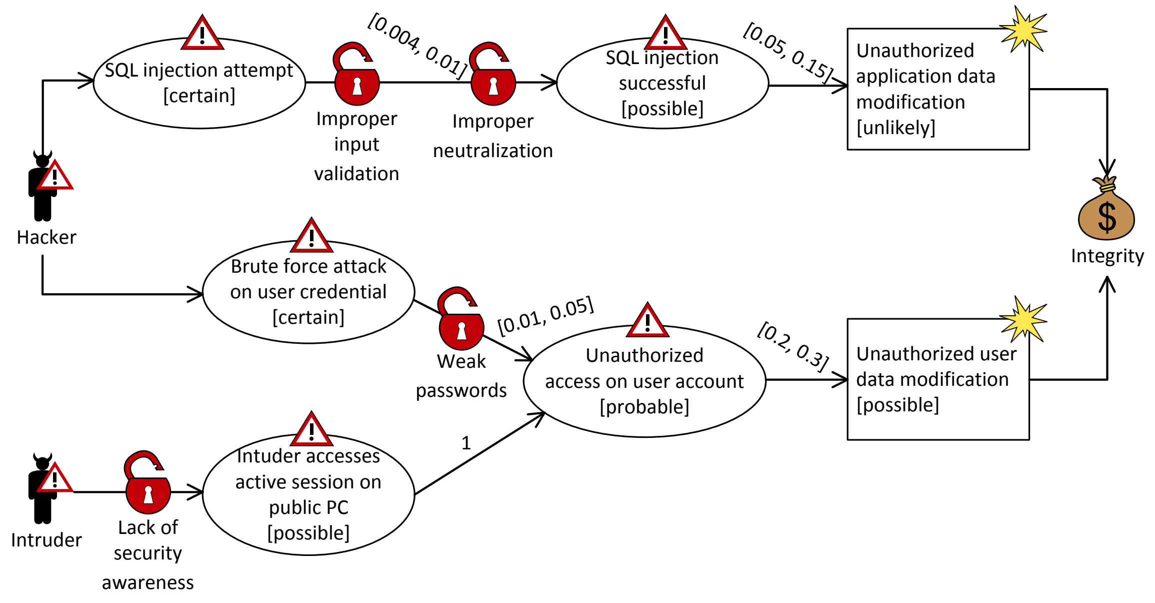

SQL injection attempt [certain]
[0.004, 0.01]
Improper input validation
Improper neutralization
SQL injection successful [possible]
[0.05, 0.15]
Unauthorized application data modification [unlikely]
Hacker
Integrity
Brute force attack on user credential [certain]
Weak passwords
[0.01, 0.05]
Unauthorized access on user account [probable]
[0.2, 0.3]
Unauthorized user data modification [possible]
1
Intruder
Lack of security awareness
Intuder accesses active session on public PC [possible]

**Figure 30 – Risk model with likelihood estimates**

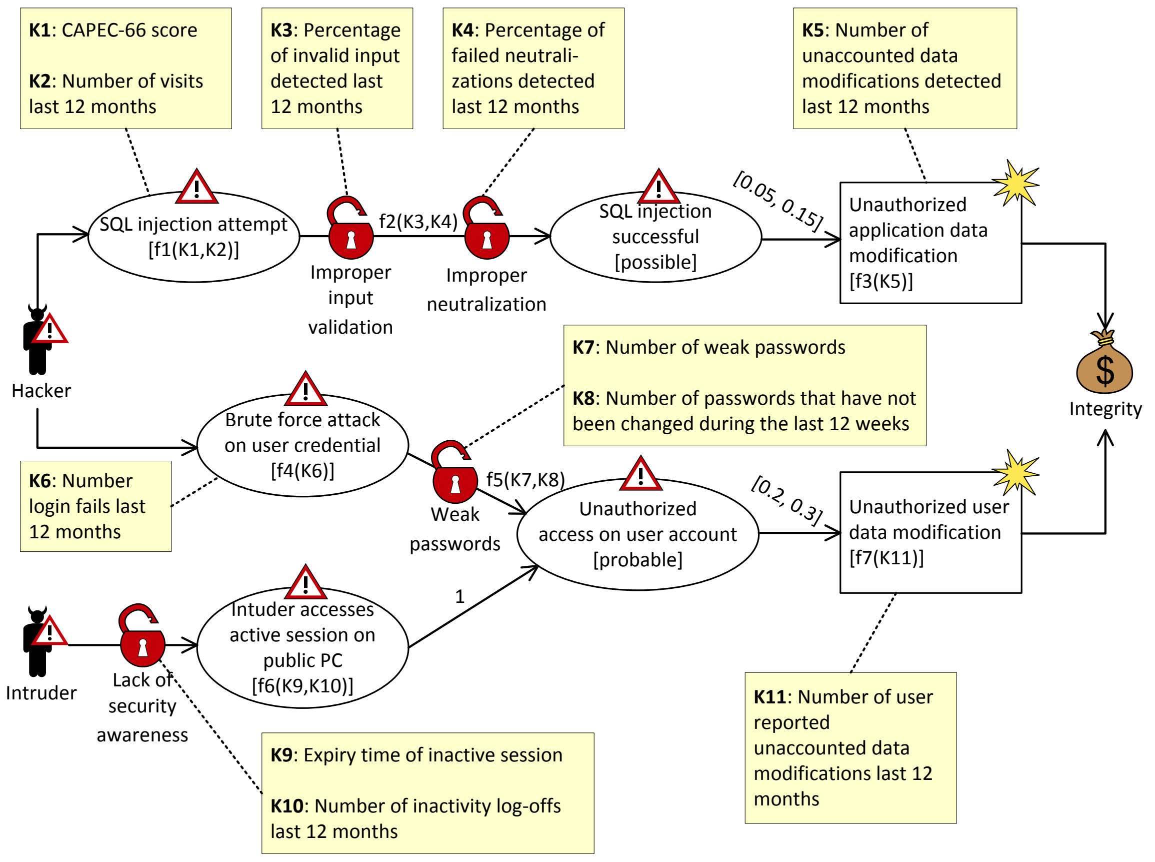
## 6.3 Key Indicator Identification

In order to monitor risks based on aggregated key indicator values we need to identify the indicators that are relevant for the risks in question. This is done by a careful walkthrough of the risk models, for example by a team of personnel with different expert insight into the target system.

Figure 31 shows a possible set of key indicators for the identified risks. In the following we explain each of them in turn.

- K1 is an indicator derived from the CAPEC attack pattern classification [12]. The pattern includes information such as required attacker skills and resources, as well as likelihood of attack. We assume here that K1 is an aggregate of such values.

- K2 is the total number of user visits on the site for the web-application, which is also related to the SQL attack since a share of the visits is the conducted attacks.
- K3 is related to the CWE [10] vulnerability of improper input validation. We assume here that the application owner conducts regular searches in their logs to keep track of the number of invalid user input.
- K4 is related to the CWE vulnerability of improper neutralization of special elements used in an SQL command (SQL injection). We assume that the application owner conducts neutralization while keeping track of the number of inputs that should have been neutralized but failed to be so.
- K5 is the number of detected data modifications that have not been accounted for by the staff of the application owner, and that therefore may include unauthorized modifications.
- K6 is the number of user login fails, which includes any attempts of brute force attacks.
- K7 is the number of weak passwords according to a classification that categorizes passwords as weak, medium strength or strong. We assume that the enforced password requirements are weak, although the users are encouraged to select strong passwords.
- K8 is the number of passwords that have not been changed during the last 12 weeks. We assume that the end-users are encouraged to change their password every three months, although this is not enforced.
- K9 is the time period of user inactivity until a session is automatically terminated. It is related to the vulnerability of lack of security awareness among end-users since this vulnerability may lead to unauthorized access to a user account due to active sessions.
- K10 is the number of forced log-offs due to user inactivity, which is also related to the possible lack of security awareness of the end-users.
- K11 is the number of user data modifications that the end-users report to the application owner as unrecognized by the end-users. This number is related to the incident of unauthorized user data modification.



K1: CAPEC-66 score
K2: Number of visits last 12 months
K3: Percentage of invalid input detected last 12 months
K4: Percentage of failed neutralizations detected last 12 months
K5: Number of unaccounted data modifications detected last 12 months
SQL injection attempt [f1(K1,K2)]
f2(K3,K4)
Improper input validation
Improper neutralization
SQL injection successful [possible]
[0.05, 0.15]
Unauthorized application data modification [f3(K5)]
Hacker
Integrity
K7: Number of weak passwords
K8: Number of passwords that have not been changed during the last 12 weeks
Brute force attack on user credential [f4(K6)]
K6: Number login fails last 12 months
f5(K7,K8)
Weak passwords
Unauthorized access on user account [probable]
[0.2, 0.3]
Unauthorized user data modification [f7(K11)]
1
Intruder
Lack of security awareness
Intuder accesses active session on public PC [f6(K9,K10)]
K9: Expiry time of inactive session
K10: Number of inactivity log-offs last 12 months
K11: Number of user reported unaccounted data modifications last 12 months

**Figure 31 – Risk model with key indicators**

Note that for risk monitoring by indicator monitoring to be valid and correct the identified key indicators must completely determine the risk estimates that they are aggregated to, possibly in combination with some given constants. The exemplified indicators are clearly not complete for the given risk model, but the purpose here is only to present the approach. However, indicators need not be identified for all risk model elements. For example, the threat scenario *Unauthorized access on user account* is not assigned any indicator. The likelihood estimate for this scenario must instead be calculated based on the likelihood estimates of the scenarios that may lead to it.

The need for the set of identified key indicators to be complete is indicated by the aggregation functions annotated in the risk model of Figure 31. For example, the likelihood of the threat scenario *SQL injection attempt* is an aggregate of the key indicators K1 and K2. The next step is to define the functions for the aggregations.

## 6.4 Specify Aggregation Functions

The aggregation functions are mappings from key indicators to likelihoods, where the likelihoods are frequencies (for scenarios and incidents) or probabilities (for leads-to relations). In Table 4 we have listed the key indicators with their domain and their current value at the time of the initial risk assessment.

| Name | Description | Domain | Value |
|---|---|---|---|
| K1 | CAPEC-66 score | {1,...,10} | 5 |
| K2 | Number of visits last 12 months | {0,...} | 120,000 |
| K3 | Percentage of invalid input detected last 12 months | [0,1] | 0.075 |
| K4 | Percentage of failed neutralizations detected last 12 months | [0,1] | 0.19 |
| K5 | Number of unaccounted data modifications detected last 12 months | {0,...} | 16 |
| K6 | Number login fails last 12 months | {0,...} | 3,200 |
| K7 | Number of weak passwords | {0,...} | 4,000 |
| K8 | Number of passwords that have not been changed during the last 12 weeks | {0,...} | 6,500 |
| K9 | Expiry time of inactive session | {5,...,45} | 15 |
| K10 | Number of inactivity log-offs last 12 months | {0,...} | 1,500 |
| K11 | Number of user reported unaccounted data modifications last 12 months | {0,...} | 25 |

**Table 4 – Key indicators with current values**

To show how the aggregation functions are specified we give only three examples. These are very simplified as the focus here is not on the functions themselves, but rather how they are used.

In our example we define the function f1 as f1(K1,K2) = (K1 · K2)/1000. This means that the number of SQL injection attempts is assumed to be proportional to the product of the total number of visits and the CAPEC score for the attack.

The function f2 is used to derive the conditional likelihood for an SQL injection attempt to lead to a successful SQL injection. The function is defined as f2(K3,K4) = (K3 + K4 – K3 · K4)/25. Generally a conditional likelihood is in the interval [0,1] but by this aggregation it is assumed that in reality it is never higher than 0.04 for this particular attack on the web-application in question.

Finally the function f3 is for the example simply defined as f3(K5) = K5/20. This means that it is assumed that a constant share of 5% of the unaccounted modifications is the unauthorized ones.

The identified key indicators and the aggregation functions should be validated. This can be done in several ways. In the following we describe two validation techniques that make use of likelihood calculation. First, we can calculate the aggregated likelihood values from the current indicator values and compare the results with the estimates from the initial risk assessment. Assuming that both are correct, they should be identical. Second, we can do a consistency check of the aggregated likelihoods; if the aggregated likelihoods are correct, they must also be mutually consistent. The first validation check can only be done when a separate risk assessment is at hand, whereas the second can be done whenever the risk picture is updated.

Considering the three specified aggregation functions and the current indicator values, we get the following results. The likelihood of the threat scenario *SQL injection attempt* is given by f1 as (5 · 120000/1000) = 600. This is within the likelihood interval of *certain* in Table 2 and matches the estimate of the initial risk assessment shown in Figure 30. The conditional likelihood of the following leads-to relation is given by f2 as (0.075 + 0.19 – 0.075 · 0.19)/25 = 0.01003. This is slightly higher than the estimate of the initial assessment which is the interval [0.004, 0.01] and may indicate that the set of identified indicators is incomplete or that the aggregation function is not correct. The likelihood of the threat scenario *SQL injection successful* is not given directly by any indicators. Instead it must be calculated using the CORAS calculus based on the monitored values that precede it. This gives the estimate 600 · 0.01003 = 6.018, which is within the *possible* interval. Also this matches the estimate from the initial risk assessment. Finally, we use the function f3 to calculate the likelihood of the incident *Unauthorized application data modification* as 16/20 = 0.8. This is within the interval *unlikely*, and again in line with the initial risk assessment.

The mutual consistency checking is to compare the monitored likelihoods against each other. For example, the current monitored likelihood of the threat scenario *SQL injection successful* is 6.018. Using the estimate of the conditional likelihood of the following leads-to relation we get the estimate 6.018 x [0.05, 0.15] = [0.3009, 0.9027] for the frequency of the unwanted incident *Unauthorized application data modification*. This is consistent with the frequency derived from the indicator K5 for this incident, namely 0.8.

The identification of the key indicators and the specification of the aggregation functions must obviously be done with much care to ensure that the result is complete and correct. The task is, however, error-prone. This means that systematic techniques for validation of the indicators and functions are important and valuable.

## 6.5 Conclusion

In this section we have presented an approach to continuous security risk assessment by means of risk monitoring. The method and techniques make use of identified key indicators that can be monitored and the values of which can be aggregated into likelihood estimates or other values that can be fed to the risk model.

The approach is similar to test-based risk assessment as presented in Section 3 and Section 4 since both approaches involves the aggregation of low-level security measures to high-level risk measures. The key indicator monitoring approach is sometimes referred to as *passive testing* whereas the security testing described in Section 3 is referred to as *active testing*. Although active and passive testing serve different purposes they are similar in the sense of using low-level measurements as input and aggregating these into useful data for the risk assessment.

# 7 Conclusion

In this deliverable we have reported on the main results of the RASEN WP3 tasks during the second year of the project. The results show progress within all of the R&D tasks of WP3 of compositional risk assessment techniques, techniques for test-based security risk assessment and techniques for continuous security risk assessment.

The developed techniques come with relevant modeling support, and they are moreover supported by the prototype tools of deliverable D3.3.2 that are integrated into the RASEN tool-box. Both WP3 and the RASEN project furthermore make strong use of available databases and repositories of known security risks and vulnerabilities, which facilitates automation as well as the development of patterns for risk identification and assessment.

## References

[1] International Electrotechnical Commission: IEC 61025 Fault Tree Analysis (FTA) (1990)

[2] International Electrotechnical Commission: IEC 60300-3-9 Dependability management – Part 3: Application guide – Section 9: Risk analysis of technological systems – Event Tree Analysis (ETA) (1995)

[3] International Organization for Standardization: ISO 31000 Risk management – Principles and guidelines (2009)

[4] International Organization for Standardization/International Electrotechnical Commission: ISO/IEC 27005 – Information technology – Security techniques – Information security risk management (2007)

[5] BSI: IT-Grundschutz-Kataloge, Bundesamt für Sicherheit in der Informationstechnik 2005-2014, English version: IT-Grundschutz Catalogues, Federal Office for Information Security, Bonn Germany (2013)

[6] W. Jansen: Directions in security metrics research. NISTIR 7564, National Institute of Standards and Technology (2009)

[7] M. S. Lund, B. Solhaug, K. Stølen: Model-Driven Risk Analysis – The CORAS Approach. Springer (2011)

[8] M. S. Lund, B. Solhaug, K. Stølen. Risk analysis of changing and evolving systems using CORAS. In Foundations of Security Analysis and Design VI (FOSAD VI), volume 6858 of LNCS, pages 231–274, Springer (2011)

[9] B. Kaiser, P. Liggesmeyer, O. Mäckel: A new component concept for fault trees. In: 8th Australian workshop on Safety critical systems and software (SCS'03), pp. 37–46. Australian Computer Society (2003)

[10] MITRE Corporation: CWE – Common Weakness Enumeration [ONLINE] Available at: http://cwe.mitre.org [Accessed 5 September 2014]

[11] MITRE Corporation: CWSS – Common Weakness Scoring System [ONLINE] Available at: http://cwe.mitre.org/cwss/ [Accessed 5 September 2014]

[12] MITRE Corporation: CAPEC – Common Attack Pattern Enumeration and Classification. [ONLINE] Available at: https://capec.mitre.org [Accessed 5 September 2014]

[13] Object Management Group. OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.2. OMG Document: formal/2009-02-02 (2009)

[14] RASEN: A toolbox for risk assessment and security testing v.2. RASEN project deliverable D5.4.2 (2014)

[15] RASEN: Methodologies for legal, compositional and continuous risk assessment and security testing v.2. RASEN project deliverable D5.3.2 (2014)

[16] RASEN: Techniques for compositional risk-based security testing v.1. RASEN project deliverable D4.2.1 (2013)

[17] RASEN: Techniques for compositional risk-based security testing v.2. RASEN project deliverable D4.2.2 (2014)

[18] RASEN: Techniques for compositional test-based security assessment v.1. RASEN project deliverable D.3.2.1 (2013)

[19] RASEN: Use case evaluation v.1. RASEN project deliverable D2.3.1 (2014)

[20] A. Refsdal, Ø. Rideng, B. Solhaug and K. Stølen. Divide and conquer - Towards a notion of risk encapsulation. In Engineering Secure Future Internet Services and Systems. LNCS 8431, 345-365, Springer (2014)

[21] W.-P. de Roever, F. de Boer, U. Hannemann, J. Hooman, Y. Lakhnech, M. Poel, J. Zwiers: Concurrency Verification: Introduction to Compositional and Noncompositional Methods. Cambridge University Press (2001)

[22] B. Solhaug and F. Seehusen. Model-driven risk analysis of evolving critical infrastructures. Journal of Ambient Intelligence and Humanized Computing, 5(2):187–204 (2014)

[23] L. M. S. Tran, B. Solhaug and K. Stølen: An approach to select cost-effective risk countermeasures exemplified in CORAS. Technical Report A24343, SINTEF ICT (2013)

[24] J. Viehmann: Reusing risk analysis results - An extension for the CORAS risk analysis method. In: 4th International Conference on Information Privacy, Security, Risk and Trust (PASSAT'12), pp. 742-751. IEEE (2012)

[25] J. Viehmann: Towards Integration of Compositional Risk Analysis Using Monte Carlo Simulation and Security Testing; in T. Bauer et al. (Eds.): RISK 2013, LNCS 8418 pp 109-119, Springer (2014)