Compositional Risk Assessment and Security Testing of Networked Systems

Deliverable D3.2.1

Techniques for Compositional Test-Based Security Assessment v.1

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:	D3.2.1
Nature of deliverable:	Report
Dissemination level:	PU
Internal version number:	1.0
Contractual delivery date:	2013-09-30
Actual delivery date:	2013-09-30
Responsible partner:	Software AG

Contributors

Editor(s)	Bjørnar Solhaug (SINTEF), Frank Werner (SAG)
Contributor(s)	Bruno Legeard (Smartesting), Fabien Peureux (Smartesting), Bjørnar Solhaug (SINTEF), Ketil Stølen (SINTEF), Johannes Viehmann (Fraunhofer), Frank Werner (SAG)
Quality assuror(s)	Erlend Eilertsen (EVRY), Bruno Legeard (Smartesting)

Version history

Version	Date	Description
0.1	13-04-15	Table of contents and document outline
0.2	13-06-25	Initial draft of all sections
0.3	13-08-26	Revision of all sections
0.4	13-09-11	Full version finalized for internal review
1.0	13-09-27	Final version

Abstract

This deliverable reports on the results of RASEN WP3 after the first year of the project. The tasks that have been addressed are (T3.1) the development of techniques for compositional security risk assessment and (T3.2) the development of techniques for test-based security risk assessment. The sections of the deliverable are structured into three main themes. The first theme is an industry-viewpoint discussion of existing approaches, and a motivation for the work presented in this deliverable. The two other themes cover research tasks T3.1 and T3.2, respectively.

Regarding compositional security risk assessment, we discuss the underlying principles of compositionality and explain how these apply to the setting of risk assessment. We furthermore present our formal foundation for compositional security risk assessment, as well as an extension of CORAS to facilitate component-based risk assessment. Regarding test-based security risk assessment, we present our approach to complement the risk picture using test results, and how we plan to use security indicators and risk metrics to aggregate low-level test results to make use of them in the more high-level risk assessment.

Keywords

Security, security risk assessment, security testing, test-based risk assessment, compositional security risk assessment

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 WP3 research tasks that were initiated at the beginning of the project, and that are covered by this deliverable, are (T3.1) the development of techniques for compositional security risk assessment, and (T3.2) the development of techniques for test-based risk identification and estimation in order to complement the risk picture based on test results.

Before reporting on the current results of these tasks, we give in Section 2 an industry-viewpoint discussion of existing approaches to motivate the work that is presented in the remained of the deliverable. The section complements the presentation of the state of the art in deliverable D3.1.1.

Section 3 to Section 5 cover the task on compositional security risk assessment. In Section 3 we discuss the underlying principles and concepts of compositionality, and how they apply to risk assessment. In Section 4 we present our formal foundation for compositional security risk assessment by defining the semantics of our risk modeling technique and introducing our operators for composing the target of analysis. The full presentation of the formal foundation is given in the appendix. In Section 5 we present our approach to component-based risk assessment with support for reusable risk analysis artifacts. The approach is an extension of CORAS, and the section also presents Strict CORAS, which is a means to facilitate reusability by ensuring that the results from different analyses are compatible.

Section 6 and Section 7 cover the task on test-based security risk assessment. In Section 6 we present our process and techniques for complementing the risk picture based on test results, where security testing is used to support both risk identification and risk estimation, and to support verification of risk treatments. In Section 7 we present our initial approach to facilitate test-based risk assessment by means of security metrics. The metrics serve as a means to aggregate low-level test results at the more high-level risk assessment. These techniques make use of both active testing (as described in Section 6) and passive testing (i.e. monitoring).

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 two directions. First, we are developing techniques 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. The current WP3 status and results of these two activities are presented in this deliverable. The activities correspond to research tasks T3.1 (compositional security risk assessment) and T3.2 (test-based risk identification and estimation), respectively, of RASEN WP3, both of which were initiated at the start of the project.

The third task of WP3, which will be initiated at M18, is the development of support for continuous risk assessment by means of indicator monitoring, as well as our techniques for compositional risk assessment. Although this task has not fully started, we present in this deliverable the initial basis for supporting continuous risk assessment.

The main RASEN research questions that we tackle in WP3 are the following:

1. To what extent can we use composition to make security assessment of large scale networked systems more feasible and understandable?
2. To what extent can we relate criteria for valid composition of security risk assessment results to criteria for valid composition of security test results?
3. To what extent can we support reuse of security risk assessments to make security assessment of large scale networked systems more feasible?
4. What are good methods and tools for aggregating test results—obtained by both active testing and passive testing (monitoring)—to the risk assessment?
5. How can test results be exploited to obtain a more correct risk picture?

The results presented in this deliverable investigate all these research questions, but the second; in order to start investigating composition across the two abstraction levels of risk assessment and testing we need to tackle the problems of compositional risk assessment and test-based risk assessment first.

The sections of this deliverable is structured into three main themes, namely an industry-viewpoint discussion of existing approaches, techniques for composing risk assessment artifacts, and techniques for test-based risk assessment. In particular, Section 2 gives a motivation of the WP3 work from an industrial perspective by reviewing existing methods, thereby addressing the first theme. The section complements the overview of the state of the art presented in deliverable D3.1.1. The second theme is addressed in the subsequent three sections. In Section 3 we discuss the underlying principles and concepts of compositionality, and how they apply to security risk assessment. In Section 4 we present our formal foundation for compositional risk assessment, and in Section 5 we present an approach to component-based risk assessment with support for composition of reusable risk analysis artifacts. The third theme is addressed in the next two sections. Section 6 presents our approach to complement the security risk picture using test results. In Section 7 we present our initial approach to facilitate test-based risk assessment by means of security metrics to aggregate low-level test results to support at the more high-level risk assessment.

The current tools and prototypes for WP3 are delivered in RASEN D3.3.1. We also refer to D5.4.1 for the initial specification of the interfaces between the various RASEN prototypes.

2 The Industrial Need for Compositional Test-Based Security Risk Assessment

There is a strong industrial need for test-based security risk assessment due to the steadily increasing complexity of today's industrial software development. Typically, such applications are distributed, logically and geographically, and encompass hundreds of installations, servers, and processing nodes. Many software systems are by now large and their interweaving with remote, networked systems is gradually gaining weight by using web services, cloud-based solutions, and secure communication. As customers rely on software systems, the products and services should not expose vulnerabilities, but reflect the state of the art technology and maintain security and technical risks at an acceptable level.

Failing to meet customer expectations would result in a loss of customer trust, customer exodus, financial losses, and in many cases in legal consequences and law suits. On the other hand, the impossibility to test and account for every potential security problem in advance is well-known within the field of security. Hence, the most critical and important security test cases must be identified.

Any security problem in such software systems could lead to a considerable damage for the customer, be it its loss of business (e.g. when a successful DoS attack prevents business processes form being pursued), loss of data (due to unauthorized access) or malicious manipulation of business process sequences or activities.

In addition to the above stated, a high level of automation is required for security risk assessment to become a success. Otherwise the huge size of tests, the analysis of the compositional artifacts, and the risk assessment of the individual components could become infeasible.

As of today there is no access for large organizations to solutions which efficiently assess security risks of their software implementations as manual processes forbid itself due to the immense complexity. Instead of following a systematic approach applied to developed software systems as proposed by RASEN, a rather ad-hoc procedure is commonly applied where security issues and risks are fixed as they are reported during the code execution. For smaller companies, having only a few products, this manual procedure may be applicable. In contrary it does not deliver promising results for large software instances as manual risk assessment does not scale when dealing with hundreds of different distinct systems. Hence, an appropriate methodology with tool support is required to improve the process for automated risk assessment and analysis on the level of software and source code. Furthermore, such a methodology should help in determining the risk level over all software applications and allow the most efficiently allocation of available resources (in terms of software developers) on the critical and most risky code fragments. As software systems constantly gain in size and reach a complexity for which easy human-eyes-only evaluation is infeasible, the process of risk and security assessment done by security experts must be supported with adequate methodologies, techniques, and tools. In addition, the successful application of such methodologies is only feasible using process automation that scales with the size of large software systems.

2.1 Currently Available Methods

There are a number of existing methods [23] in the area of risk assessment. Still, it is quite hard to identify those which at least partially support the IT risk management process relevant for secure and compositional industrial application development. In the remainder of this section we present and discuss existing methods for security risk analysis and the set of methods is not limited as there is still very active ongoing research.

All methods have the same fundamental target, but attempt to hit the target from very different approaches as some are applicable to all types of risk while others are restricted to specific risks. In this sense, the following summary will highlight methods which are scalable and applicable to large scale systems and consider the degree to which methods can be automated with respect to the model creation and testing. As there is a strong urge to apply the methods also to real world scenarios, software security is also considered.

Strengths and weaknesses of each methodology are pointed out, before eventually a comparison is given. The overview complements the presentation of the state of the art in RASEN deliverable D3.1.1.

2.1.1 The CORAS Method

The CORAS method [32] provides a graphical language for threat and risk modeling, and a model based method for security risk analysis. It includes a diagram editor and detailed guidelines explaining how the language should be used.

CORAS has been developed through both empirical investigations and a series of industrial field studies, and it is closely based on established international standards, in particular the ISO 31000 risk management standard (projects financed by the Norwegian Research Council and the EU). It is based on international standards for risk management [11].

Benefits

The CORAS language is also perceived as easy to learn and understand based on the users' familiarity with other modeling languages (participant language knowledge appropriateness). Its resemblance with use cases and fault trees makes it easy to understand for new users, even though they are not familiar with the notation. Due to its already existing tool support and its formal, textual syntax, it has a high appropriateness factor for computerized tools (technical actor interpretation appropriateness) [8].

Weakness

In the CORAS method, aspects that still can be further improved, mainly related to comprehensibility appropriateness, the language's ability to express all security related scenarios (knowledge externalizability appropriateness) and its appropriateness for various types of organizations (organizational appropriateness). Another problem is the usability of the diagrams: when the CORAS diagrams grow in size they can easily become cluttered and difficult to read.

Since it is highly recommended to document the findings from the brainstorming immediately, preferably by modeling it "on-the-fly", it sets high demands to the analysis secretary. However, the CORAS tool is designed to support these activities, and the tool ensures the syntactical correctness of the diagrams.

The current version of the CORAS approach provides little support for combining results from different analyses (for instance, analyses of two components of the same system). This kind of modularity might for example reduce the effort related to keeping an analysis documentation updated, since only the part where a change has been made needs to be reanalyzed. See Section 3 for further details on this challenge.

2.1.2 The COBRA Method

COBRA[1], or "Consultative, Objective and Bi-functional Risk Analysis", consists of a range of risk analysis, consultative and security review tools. COBRA Risk Consultant provides a complete risk analysis service, compatible with most recognized methodologies (qualitative and quantitative). It is a questionnaire based PC system using 'expert' system principles and an extensive knowledge base. The default process usually consists of three stages:

1. Questionnaire Building
2. Risk Surveying
3. Report Generation

During the first stage, via module selection or generation, the base questionnaire is built to fit the environment and requirements of the user. The second stage is the survey process - Risk Consultant questions are answered by appropriate personnel and the information is securely stored. For the third stage risk assessments and 'scores' are produced for individual risk categories, individual recommendations are made and solutions offered, and potential business implications are explained[1].

[1] http://www.security-risk-analysis.com/

Benefits

It was recognized that business users should be involved from the outset. This carries a number of advantages, and shapes the entire review. In addition, a number of other radical departures were called for. The risk assessment process, using COBRA, is extremely flexible. A substantial number of approaches are supported. Other benefits are the flexibility, automatic customization, self- analysis, reports and solution testing included in COBRA.

Weakness

COBRA is based on the various questionnaire or survey i.e. opinion based; the participants may or may not be well aware with the recent developments in the concerned area. This methodology is a generalized one; hence, there is still a need to develop or extend the methodology for particularly requirements phase. Also, the accuracy level of this methodology is also not mentioned. Risk assessment technique is not clearly mentioned. COBRA does not clearly talk about the security attributes e.g. Confidentiality, Integrity, and Availability etc. Threats and vulnerabilities play a very important role in the process of risk assessment; but how these are taken into consideration, is not clearly given in the methodology [23].

2.1.3 The CRAMM Method

CRAMM[2] is based on the UK Government's preferred risk assessment methodology. It is a total information security toolkit that includes:

- A comprehensive risk assessment tool that's fully compliant with ISO 27001
- A range of help tools to support information security managers to plan and manage security
- Wizards to rapidly create pro-forma information security policies and other related documentation
- Tools that support the key processes in business continuity management
- A database of over 3000 security controls referenced to relevant risks and ranked by effectiveness and cost
- Essential tools to help achieve certification or compliance to ISO 27001.

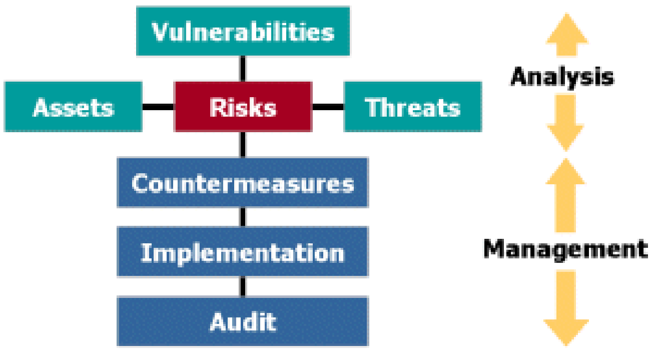


Figure 1 – Overview of the CRAMM method

Benefits

CRAMM provides a staged and disciplined approach embracing both technical (e.g. IT hardware and software) and non-technical (e.g. physical and human) aspects of security. The risk assessment tools are extremely flexible and allow you to explore different issues and answer many different questions. CRAMM contains a variety of tools to help evaluate the findings of a risk assessment including:

- Determining the relative priority of controls
- Recording the estimated costs of implementing the controls
- Modeling changes to the risk assessment, using 'what-if?' calculations

[2] http://www.cramm.com/

- Back-tracking through the risk assessment to show the justification for specific controls

Weaknesses

The weakness of the CRAMM method is that qualified and experienced practitioners are needed to handle the tool. In addition, full reviews may last long and produce too much hard-copy output which may be minimized by keeping the analysis at a required minimum. Due to the delay between the analysis and the implementation some results of a full review are possibly insignificant, in particular after rapid changes to system or networked review [39].

This methodology is a generalized one; hence, there is still a need to develop or extend the methodology for particularly requirements phase. Quantitatively risk assessment cannot be provided by CRAMM. For a list of vulnerabilities, the source is not clearly mentioned. CRAMM does not clearly talk about the security attributes e.g. Confidentiality, Integrity, and Availability etc. 'How the severity of threats and vulnerabilities is mapped', is not clearly given in CRAMM. Hence, there is a need to re-look in this perspective [23].

2.1.4 The OCTAVE Method

OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) [1] is a suite of tools, techniques, and methods for risk-based information security strategic assessment and planning. There are three OCTAVE methods[3]:

- the original OCTAVE method, which forms the basis for the OCTAVE body of knowledge
- OCTAVE-S, for smaller organizations
- OCTAVE-Allegro, a streamlined approach for information security assessment and assurance

OCTAVE methods are founded on the OCTAVE criteria - a standard approach for a risk-driven and practice-based information security evaluation. The OCTAVE criteria establish the fundamental principles and attributes of risk management that are used by the OCTAVE methods.

When applying OCTAVE, a small team of people from the operational or business units and the IT department works together to form the analysis team and addresses the security needs of the organization. The analysis team

- Identifies critical information assets
- Focuses risk analysis activities on these critical assets
- Considers the relationships amongst critical assets, the threats to these assets and the vulnerabilities (both organizational and technological) that can expose assets to threats
- Evaluates risks in operational context, i.e., how the critical assets are used to conduct the organization's business and how they are at risk due to security threats and vulnerabilities
- Creates practice-based protection strategy for organizational improvement as well as risk mitigation plans to reduce the risk to the organization's critical assets

Benefits

The OCTAVE methods are

- self-directed—Small teams of organizational personnel across business units and IT work together to address the security needs of the organization, meaning that people from within the organization assume responsibility for setting the organization's security strategy.
- flexible—Each method can be tailored to the organization's unique risk environment, security and resiliency objectives, and skill level.
- evolved—OCTAVE moved the organization toward an operational risk-based view of security and addresses technology in a business context.

Unlike the typical technology-focused assessment that is targeted at technological risks and focused on tactical issues, OCTAVE is targeted at organizational risk and focused on strategic, practice-related issues. It is a flexible evaluation that can be tailored for most organizations[3].

Weakness

Risk evaluation criteria are based on a qualitative scale (high, medium, low). This methodology is a generalized one; hence, there is still a need to develop or extend the methodology for particularly requirements phase. OCTAVE considers only three attributes for risk assessment: Confidentiality, Integrity, and Availability. There are some other attributes which may also be taken into this list for risk calculation factors. This will improve the accuracy of the risk assessment. What is the accuracy level of this methodology is also not mentioned. The methodology is much opinion based; the participants of the workshop may or may not be well aware with the recent developments in the concerned area. Hence, there is a need to quantify a maximum no. of steps [23].

Experts say one of the drawbacks of OCTAVE is its complexity. "There was a lot of time taken up just trying to understand what the approach was, because it wasn't very clear. A downside to OCTAVE is that it doesn't allow organizations to model risk mathematically or quantitatively.

2.1.5 The FAIR Method

Factor analysis of information risk (FAIR for short) [25] provides taxonomy of the factors that contribute to risk and how they affect each other. It is primarily concerned with establishing accurate probabilities for the frequency and magnitude of loss events. It is not, per se, a "cookbook" that describes how to perform an enterprise (or individual) risk assessment.

FAIR seeks to provide a solid understanding of what risk is, what the factors are that drive risk, and a standard nomenclature, as well as a framework for performing risk analyses. Much of the FAIR framework can be used to strengthen, rather than replace, existing risk analysis processes like those mentioned above. FAIR is not another methodology to deal with risk management, but it complements existing methodologies.

Benefits

Where other risk assessment standards focus their output on qualitative color charts or numerical weighted scales, the FAIR model specializes in financially derived results tailored for enterprise risk management. FAIR has been widely accepted and used within the Finance, Government, Healthcare, and Retail industries[4].

Weakness

FAIR can be difficult to use and it's not well documented. Hayes cites as a shortcoming of FAIR the lack of access to current information about the methodology and examples of how the methodology is applied [37].

- Concerns tend to revolve around one or more of the following issues:
- The absence of hard data.
- The lack of precision.
- It takes some of the mystery out of the profession. The fact is, there are those who prefer to be artists – in some cases because an artist can never be judged as "wrong."
- FAIR analysis appears to be hard work
- FAIR appears complicated

2.1.6 The NIST Risk Management Framework

The 6-step chart below can be used to link to FIPS, SP's, FAQ's and Quick Start Guide documents for the RMF steps.

[3] http://www.cert.org/octave/
[4] http://www.cxoware.com/what-is-fair/

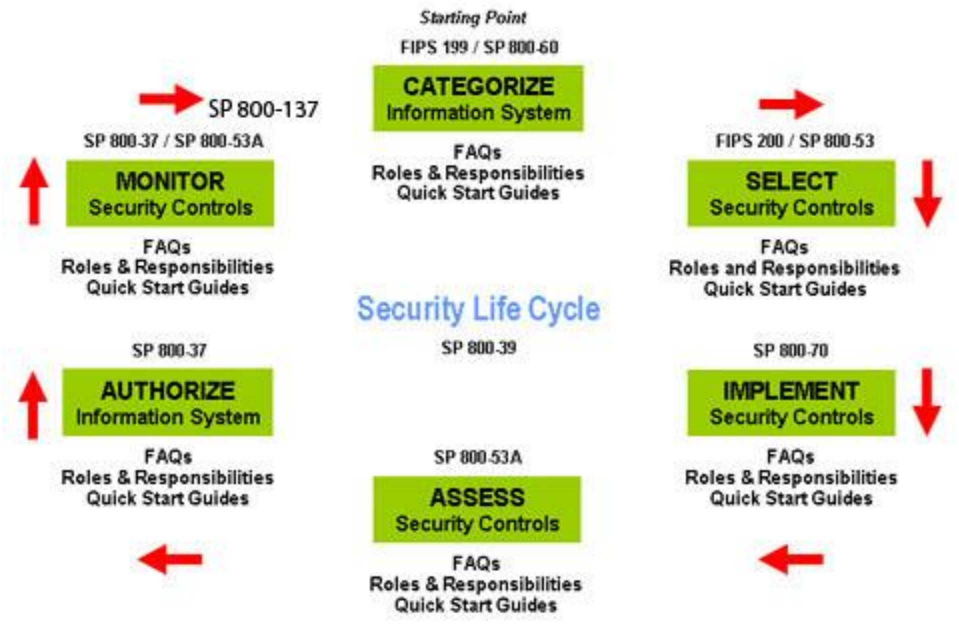


Figure 2 – Security Life Cycle according to the NIST Method

The Risk Management Framework provides a structured, yet flexible approach for managing the portion of risk resulting from the incorporation of information systems into the mission and business processes of the organization. The risk management concepts are intentionally broad-based with the specific details of assessing risk and employing appropriate risk mitigation strategies provided by the supporting NIST security standards and guidelines[5].

Benefits

The Risk Management Framework (RMF) provides a disciplined and structured process that integrates information security and risk management activities into the system development life cycle.

One of the primary strengths of RMF is that it was developed by the NIST, which is charged by Congress with ensuring that security standards and tools are researched, proven and developed to provide a high level of information security infrastructure. Because government agencies and the businesses that support them need their IT security standards and tools to be both cost-effective and highly adaptable, the framework is constantly being reviewed and updated as new technology is developed and new laws are passed. Furthermore, independent companies have developed tools that support the NIST standards, knowing that the basis for applications is stable; software development companies are more willing to develop application tools to support the framework. The model also helps companies determine when something exceeds a certain threshold of risk[6].

Weakness

NIST RMF

As for weaknesses, like any of these frameworks, you have to make sure that the people who are doing the risk assessment have the discipline to put reasonable data into the model so you get reasonable data out.

Also, it's a document; it's not an automated tool. Another weakness of RMF is its nomenclature. The use of acronyms throughout the framework and supporting tools is pervasive [37].

NIST GUIDE:

This methodology is a generalized one i.e. for all the major phases of SDLC; hence, there is still a need to develop or extend the methodology for particularly requirements phase. The likelihood of the vulnerabilities is described as high, medium, or low; but at what basis, these levels are allocated, is not

[5] http://csrc.nist.gov/groups/SMA/fisma/Risk-Management-Framework/
[6] http://itsecurityoffice.blogspot.de/2011/09/nist-rmf-national-institute-of.html

clearly mentioned in the report. For threat sources, all types of threats are taken into consideration. But from security perspective, some threats like natural threats e.g. floods, earthquakes, tornadoes, landslides etc. are not much relevant. For list of vulnerabilities, source is not clearly mentioned. NIST does not talk about the quantification of the risk. Although some calculation is done, it is not accurate; because the values of high, medium, and low impact is fixed as 100, 50, and 10.

In the step 3 of the report i.e. Vulnerability Identification, there is a step System Security Testing which cannot be followed at the requirements level. Hence, for requirements level risk assessment, the methodology may be revisited. Impact analysis is performed on the basis of three attributes: confidentiality, integrity, and availability. There are some other attributes like authenticity, non-repudiation which may also be taken into this list for performing impact analysis [23].

2.1.7 The SOMAP Method

One of the main goals of the Security Officers Management and Analysis Project (SOMAP)[7] is to develop and maintain Open Source Information Security Risk Management documents, tools and utilities.

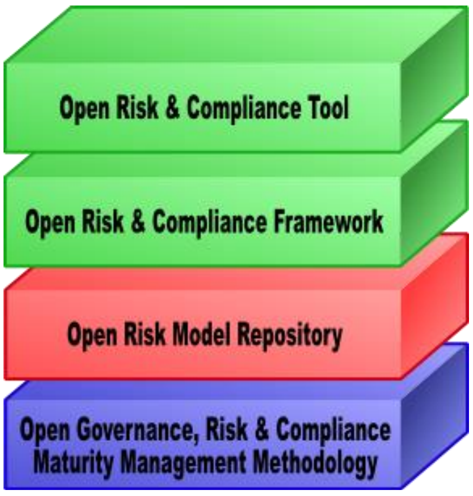

Open Risk & Compliance Tool
Open Risk & Compliance Framework
Open Risk Model Repository
Open Governance, Risk & Compliance Maturity Management Methodology

Figure 3 – Different layers of the DOMAP Method

The activities are concentrating on four sub-projects

- The OGRCM3 project develops and documents a methodology on how to measure and manage risk.
- The ORIMOR contains a database model which is used as the basis for our own risk management framework and tool.
- The ORICO Framework and Tool are the (reference) implementation of our own maturity management methodology.

They understand the risk management process as a lifecycle with four steps, starting at the Compliance Scoping.

- Compliance Scoping
  - In this step you define what authority documents you will need and for which you try to achieve compliance with.
- Asset Management & Categorization
  - During this step you manage your assets, you define the responsibilities and manage the changes since the last assessment.
- Compliance Measurement & Documenting

[7] http://www.somap.org/

  - This step is about measuring the compliance of the implemented controls with the requirements as described in the authority documents chosen during the Compliance Scoping step.
  - Different assessment measuring strategies can be used to measure the compliance level. Findings are documented for later evaluation and reporting.
- Evaluation & Reporting
  - The last step contains the evaluation of the findings and the reporting of the facts to the upper management and other interested parties.

Benefits

SOMAP is freely available and released under an open source license. Other benefits are the following:

- Community: Everybody interested can become a part of. And all the contributions made to the project are made freely available so others can benefit as well.
- Independence: SOMAP is not on the payroll of anybody and therefore we can do what is best. Our Repository contains no commercials but the straight facts and information needed.
- Best of Breed: Since we are independent we - in every situation - use what's best for the project. Like that our SOBF Management Tool is based on Java, which is a widely accepted industry standard.
- Trust: We have nothing to hide (and because we trust in Open Source, we would have no chance to do so anyway). Our data and the source to our projects are freely available and can be scanned before they are used. This proceeding guarantees that potential problems within our design or source can be detected and everybody can address them.
- Competence: Since the project is community based, our results get peer reviewed. Potential problems can be addressed and solved in a discussion. And best of all, everybody is welcome to help out and help the project to improve its standard.

Weakness

This methodology is a generalized one; hence, there is still a need to develop or extend the methodology for particularly requirements phase. The methodology considers five key attributes for risk assessment: Confidentiality, Integrity, Availability, Accountability, and Auditability. There are some other which may also be taken into this list for risk calculation factors. This will improve the accuracy of the risk assessment.

The method talks about the 'Cost of Control', but it is not mentioned how this factor will be calculated. On which basis all the ranks or values of components are defined is not mentioned in the report. What is the accuracy level of this methodology is also not mentioned. Therefore, one may validate this methodology and discuss the results by applying the same. Threats and vulnerabilities play a very important role in risk assessment process. Although the method considers both the things in the beginning, but in the calculation part, only likelihood and impact of vulnerabilities are taken into consideration [23].

2.1.8 The IA OM Method

Ting and Comings [5] described how to use the Information Assurance (IA) Object Measurement (OM®) metric as a tool to measure the monitoring step described in the United States (U.S.) National Institute of Standards and Technology's (NIST) Risk Management Framework (RMF)1. The metrics or results derived from the IA OM® methodology are meaningful to the organization because the measurements themselves are "tied directly to questions that are important to the organization". The results are also useful to organizational management since they indicate the degree to which specific information security risk management goals are being met as action is taken to improve an organization's overall information security posture in terms of its information security objectives . In this instance IA OM® can be conceptually expressed as providing a measure of the degree to which the organization's information security risk management objectives are being met.

Benefits

The IA OM® metric is a good choice for use as an enterprise risk management metric, due to its versatility as a management metric. This metric:

- Measures information security risk management activities within an organization;
- Shows organizational senior management where their organization currently stands with respect to its risk management strategy, and its monitoring plan; and
- Demonstrates to senior management how such metrics can be used over time to track and improve their organization's ability to meet its overall risk management strategy and risk monitoring plan.
- The metrics or results derived from the IA OM® methodology are meaningful to the organization because the measurements themselves are "tied directly to questions that are important to the organization".

Weakness

IA OM needs manual work to fulfill the steps in risk management process. Also, the answers to the questions depend on the individual knowledge of the expert.

2.1.9 The LAVA Method

LAVA stands for the Los Alamos Vulnerability/ Risk Assessment system and is a risk methodology developed by the Los Alamos National Laboratory [38]. It is a systematic methodology for assessing vulnerabilities and risks in complex safeguard and security systems. LAVA was developed to address large, complex systems that are generally too large for other risk analysis methods.

[They] have developed an original methodology for performing risk analyses on subject systems characterized by a general set of asset categories, a general spectrum of threats, a definable system-specific set of safeguards protecting the assets from the threats, and a general set of outcomes resulting from threats exploiting weaknesses in the safeguards system. The Los Alamos Vulnerability and Risk Assessment Methodology (LAVA) models complex systems having large amounts of "soft" information about both the system itself and occurrences related to the system. Its structure lends itself well to automation on a portable computer, making it possible to analyze numerous similar but geographically separated installations consistently and in as much depth as the subject system warrants. LAVA is based on hierarchical systems theory, event trees, fuzzy sets, natural-language processing, decision theory, and utility theory.

Benefits

Personnel at the subject site see only an interactive questionnaire eliciting from them data about the presence and quality of the safeguards, the potential consequences of a successful threat, and the target organization's preference structure—the technical expertise is built into the model (and the computer code) itself. LAVA yields quantitative and qualitative insights: a pair of values (monetary and linguistic) express loss exposure for each threat/asset/safeguards-function/outcome quadruple. Using LAVA, we have modeled our widely used computer security application as well as LAVA/CS systems for physical protection, transborder data flow, contract awards, and property management. LAVA's approach is technically sound, accurate, interactive, secure and portable.

Weakness

LAVA does only consider the following threads:

- natural and environmental hazards
- accidental and intentional on-site human threats (including the authorized insider)
- off-site human threats

2.2 Methods in the Context of Industrial Software Systems

As one can see, all methods do need manual work to fulfill the risk analysis. There is no tool which is fully automatic. Many tools provide semi-automatic structures with questionnaires and automatic analysis afterwards.

All presented methods are used and proved in research or/as well as in industrial projects. But for the goals and objectives of the RASEN project and the identified industrial needs, they are not fully adequate. To complete the research task, a security risk analysis method is needed which is scalable to large systems, sufficiently automated, fast, applicable to software security, usable with real software and should incorporate automated testing.

An overview of how the presented methods meet these requirements is depicted in Table 1. The overview matrix reads in the following way: "+" states that the feature is supported, "-" that it is not supported, "o" indicates a partly support, and fields with "?" indicate the support is unknown.

Method	Scalable	Automatic	Fast	Applicable to software security	Usable with real software	Automated testing
CORAS	o	Model must be created manually	?	-	-	-
COBRA	+	Questionnaire must be answered manually	o	o	+	o
CRAMM	o	Tool must be used manually	?	+	+	-
OCTAVE	o (OCTAVE Allegro)	Analysis team must do manual work	-	o	-	-
FAIR	o	-	?	-	+	-
RMF	o	-	-	+ (There exist tools developed by independent companies)	+	-
SOMAP	?	-	?	+	+	?
IA OM	-	-	?	-	?	-
LAVA	+	Questionnaire must be answered manually	?	LAVA does only consider natural and environmental hazards, accidental and intentional on-site human threats, off-site human threats	+	-

Table 1 – Comparison of Existing Method

Basically, the methods discussed above have in common that they only partially satisfy the identified requirements. Although useful, they are not fully adequate in the context of large-scale and security critical software systems. In particular they lack scalability of the methodology to cover thousands of libraries and application to large-scale systems has never been feasible on a coordinated and fully-automated approach.

As discussed in our state of the art and RASEN baseline deliverable D3.1.1, there is currently no or very limited support for understanding and assessing security risks in a compositional manner. A systematic and efficient approach to compositional security risk assessment could significantly contribute to solve the scalability issue. A further open question is how testing could be incorporated to facilitate and improve the security risk assessment.

Software AG currently uses a variant of the FAIR methodology customized in collaboration with the Fraunhofer SIT institute for application to the large software product development teams for the top-down product security risk analysis and a bespoke analysis system for the bottom-up data sources aggregation. RASEN aims to improve the state of the art for all properties shown in Table 1 which currently prevent an efficient application of security risk assessment on large scale software applications in an automated fashion. In the remainder of this deliverable we present the first steps of RASEN WP3 in developing and providing support for compositional security risk assessment (Sections 3 to Section 5) and support for test-based security risk assessment (Section 6 and Section 7)

3 Composition and Compositionality in Risk Assessment

Risk assessment is about understanding and documenting what can go wrong for a given target of analysis, how this may happen, how likely it is and how much harm it causes. In the following we use *T* to denote the target of analysis and *R* to denote the risk documentation. We often refer to the latter as risk model.

The target of analysis is the actual system that we wish to have analyzed. Hence, the target includes all possible behaviors and actors of the system. Appropriately understanding the target requires many different aspects to be taken into account, such as users and roles, work and business processes and policies, software and hardware systems, premises, environment, and so forth. The risk model that we develop during a risk assessment can be understood as an abstraction of the target where we specify only the behavior that is relevant for understanding the risks, and at suitable level of details.

As such, risk modeling is an important risk assessment technique as it is a means for capturing, understanding and analyzing precisely the parts or aspects of the target that we are interested in from a risk management perspective. Given *R* and *T*, a very crucial issue in any risk assessment is therefore the correctness of the former. In other words, in any risk analysis we need somehow to prove or substantiate that *R* is a correct specification of *T*.

For most risk analyses, only certain aspects or parts of the target are analyzed. This is because it is usually infeasible or too time and resource consuming to conduct one analysis of the whole system at the same time. In such analyses we can make use of existing techniques [14][15] to verify or substantiate the correctness of *R* with respect to *T*. This is fine as long as we can sufficiently understand the risks of the target of analysis by investigating the different parts separately. However, for certain large-scale and complex systems, we can properly and adequately understand the full risk picture only by considering all parts and aspects together and in combination. Considering the infeasibility of assessing the whole target at once, we need techniques for systematically composing separate risk assessment results to derive the full picture.

More precisely, such an approach involves decomposing the target *T* into separate parts or aspects $T_1,...,T_n$ and analyzing each of them separately to build the risk models $R_1,...,R_n$, respectively. For each such part we use traditional techniques to verify the correctness of each R_i with respect to T_i, $i \in \{1,...,n\}$. Novel techniques using composition are then required for deriving the correct risk model *R* based on $R_1,...,R_n$ and the composition of $T_1,...,T_n$.

In the following we discuss the underlying concepts and principles of such use of composition. First we discuss composition in general, and then the principles of compositionality.

3.1 Composition

Composition is widely used in system engineering and analysis, and in different disciplines such as business modeling, requirements engineering, architecture, design and implementation. Specification languages come with composition constructs, such as AND/OR composition of goals in goal-oriented requirements engineering [21], parallel and sequential composition in UML diagrams [22], and AND/OR gates in fault trees [7]. In making specifications, composition is a feature that facilitates the specification of different parts separately with various means to combine them in order to capture the desired overall specification.

Within risk assessment, composition can be utilized in order to systematically manage the assessment of large-scale and complex systems. This requires techniques and guidelines for how to systematically derive the combined risk model *R* for a target *T* given separate analysis results R_i for individual parts T_i of the target. The challenge in such an approach is to carefully take into account the potential dependencies that may exist between the different parts of the target. Such dependencies can affect the overall risk picture in unforeseen ways, and need to be analyzed and resolved while combining the results. For example, when combining services of T_1 and T_2, we may increase redundancy to the extent that previously unacceptable risks of availability become acceptable. Conversely, service composition may introduce new threats due to the combination of different vulnerabilities.

For approaches that are not compositional in the sense discussed below, the systematic use of composition in risk assessment therefore requires further analyses of the target and the risks when they are composed. For such approaches to of value, the time and resources to conduct a full analysis from scratch for *T* should be (significantly) more than the total effort of the composition approach.

3.2 Compositionality

Our notion and principles of compositionality in risk assessment is based on the notion of compositionality in program verification as established in the formal methods community. In this setting, de Roever et al. define compositionality as follows: "That a program meets its specification should be verified on the basis of specifications of its constituent components only, without additional needs for information about the interior construction of those components" [24].

In risk assessment we are concerned about the correctness of the risk mode *R* with respect to a target *T*. The principle of compositionality in this setting therefore means the following: That a risk model is correct should be verified on the basis of separate parts of the target only, without additional needs for information about the interior construction of these parts.

A compositional approach to risk assessment must be founded on a compositional proof method that basically consists of two parts, namely a basic technique and a compositional proof technique. The basic technique is for proving that a risk model R_i is correct for a target T_i that is not decomposed further. The compositional proof technique is for handling the case that *T* is composed of the parts $T_1,...,T_n$ for a given composition construct. For such composition constructs, the challenge is to develop compositional proof rules of the following form: From R_1*is correct with respect to*T_1 and ... and R_n *is correct with respect to* T_n, conclude that *R is correct with respect to T*.

The crucial feature about compositionality is that the application of the proof rule shall not require any reference to or further investigation of the internal construction of the different parts T_i of the target. This shall be completely left to the application of the basic technique (i.e. traditional risk assessment techniques). However, the applicability and soundness of the compositional proof rules generally rely on some additional conditions that must be satisfied. To obey the principles of compositionality, these conditions can only be upon the statements about the target as captured by the risk model.

More formally, a compositional approach to risk assessment requires the development of compositional proof rules of the following form for some composition construct ⊕.

$$\frac{\begin{array}{c}\text{for } i = 1,...,n\text{: } R_i \text{ is correct with respect to } T_i \\ R \text{ follows from some combination of } R_1,...,R_n \\ [\text{additional conditions upon } R_1,...,R_n]\end{array}}{R \text{ is correct with respect to } \oplus(T_1,...,T_n)}$$

In order to develop such rules, there are a number of issues that must be addressed. How should the target *T* and the risk model *R* be specified and formalized? How should the notion *R is correct with respect to T* be captured and formalized? What kind of composition and decomposition of the target can we support? What kind of additional conditions do we need when defining the compositional proof rules?

In developing an approach to compositional risk assessment we need to build a formal foundation that serves as a rigorous basis for tackling issues like these. Such a foundation should provide the means for formal proofs of the soundness of the compositional proof rules. These proofs are our techniques as method developers to ensure and demonstrate the soundness of the approach. For risk analysts using the approach, i.e. the end-users, the applicability of the compositional proof rules requires the premises to hold. This includes premises such as R_i *is correct with respect to* T_i when T_i is not decomposed further. As mentioned above, this must be substantiated using existing and traditional risk assessment methods and techniques. An important practical difference from formal methods is that while the latter seek to formally prove that a program meets its specification, this can in the general case not be expected in risk assessment. Risk assessment rather uses empirical methods, triangulation, testing, etc. to substantiate that the premises hold. The compositional proof rules should then be applicable when the analyst has sufficient confidence that the premises hold.

4 Foundations for Compositionality in Risk Assessment

The aim of this section is to establish an initial formal foundation for compositionality in risk assessment. While respecting the principles of compositionality as discussed in Section 3, the objective is to introduce the basic means for capturing and reasoning about the target of analysis and the risks, and to introduce notions of target composition. The full formalization is presented in the appendix. We also refer to two recent publications for details about the formal semantics [35][36].

We assume a model-driven approach in which the risk model is represented in terms of risk graphs [3].A risk graph can be understood as a common abstraction of several established risk modeling techniques (including Bayesian networks [1], fault trees [7], event trees [10] and CORAS threat diagrams [20]). Our approach is to develop techniques and rules that apply to risk graphs and that can be instantiated in other risk modeling techniques, thereby ensuring a more general applicability than by developing compositional techniques for one specific risk modeling language only. While we do not assume any specific modeling technique for representing the target of analysis, we represent it formally by a set of execution traces. These traces represent possible execution histories, and may represent an existing running target or a (model-based) specification of the target, for example a UML [22] model.

4.1 Representing the Target of Analysis

The target of analysis is represented by a set of traces H, where each $h \in H$ represents an execution history. A history is an infinite sequence of timed events that is ordered by time and progresses beyond any finite point in time. $\mathbb{E}$ denotes the set of all events, while the set of all timestamps is defined by $\mathbb{T} \overset{\text{def}}{=} \mathbb{R}^{+}$, where $\mathbb{R}^{+}$ denotes the set of non-negative real numbers. A timed event (e, t) is an element of $\mathbb{E}$ x $\mathbb{T}$. Hence, the set of all histories $\mathbb{H}$ is the subset of $(\mathbb{E} \text{ x } \mathbb{T})^{\infty}$ in which the events are ordered by time, and where time always progresses beyond any finite point in time.

In order to specify and reason about risk we need to represent likelihoods. In earlier formalizations of risk graphs we used probabilities to capture likelihoods, and we developed a calculus for probability reasoning in risk graphs [3]. In our ongoing work we have introduced frequencies as an alternative to probabilities, and developed a calculus for the sound reasoning about frequencies, all of which is presented in the appendix. Operating with probabilities is useful in order to support formal reasoning because of the well-defined probability theory. However, when conducting security risk assessment in practice, probabilities can be hard to obtain and are often difficult for stakeholders to relate to.

To capture frequencies we set the time unit as equal to 1, and assume for simplicity that all frequencies are per time unit. The set of frequencies $\mathbb{F}$ is defined by $\mathbb{F} \overset{\text{def}}{=} \mathbb{R}^{+}$. This means that $f \in \mathbb{F}$ denotes the frequency of f occurrences per time unit.

4.2 Risk Graphs

Risk graphs are a risk modeling technique for specifying how scenarios may lead to other scenarios, where a scenario represents a threat scenario or an unwanted incident. Scenarios are represented by vertices, and a relation from one vertex to another means that the former may lead to the latter. A vertex can be assigned a frequency *f*, which is a specification of the likelihood of the scenario to occur. A frequency can also be assigned to relations between vertices to capture the statistical relationship between them, i.e. the extent to which the occurrence of one scenario will lead to another. An example of the risk graph notation is depicted in Figure 4.

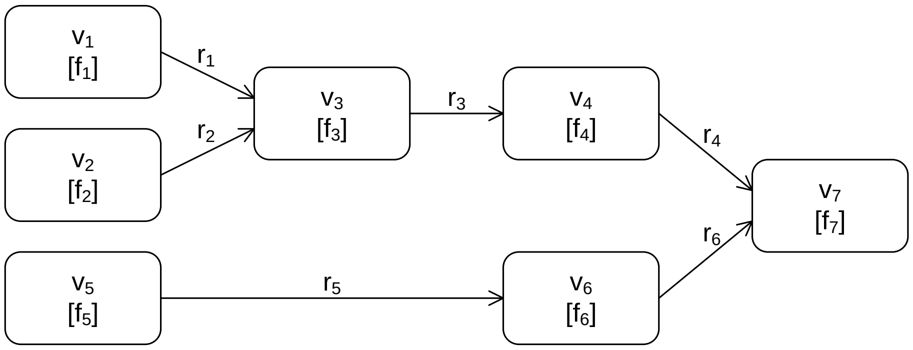


Figure 4 – Risk graph

Formally, a risk graph is a pair of two sets (V, R) where $V \subseteq \mathbb{P}(\mathbb{E}) \times \mathbb{F}$ and $R \subseteq V \times \mathbb{R}^+ \times V$. For any set of elements A, $\mathbb{P}(A)$ denotes the powerset of A. We use $v(f)$ to denote a vertex and $v \to^r v'$ to denote a relation.

A risk graph formula is the form $H \vdash v\ (f)$ or $H \vdash v \to^r v'$. Hence, a risk graph formula is always with respect to a given target of analysis as represented by the trace set H, which is required to be non-empty. Semantically a vertex formula of the form $H \vdash v(f)$ is a statement about the frequency of the occurrences in H of the events in v, while a formula of the form $H \vdash v \to^r v'$ is a statement about the statistical relationship between the events in v and v' as expressed by r. For a given H, the semantics of a risk graph formula is therefore one of the Boolean values *True* and *False*.

A risk graph specification is of the form $H \vdash (V, R)$. The semantics is the conjunction of the semantics of each risk graph formula in V and R.

Risk graphs also come with a calculus with rules for reasoning about likelihoods and checking the consistency of the risk models. The calculus is presented in the appendix, where the risk graph syntax, semantics and calculus are also generalized to capture countermeasures and consequences, as well as intervals of likelihoods and consequences instead of restricting the expressiveness to exact values only.

4.3 Basic Techniques

In compositional verification, basic techniques are for proving that a program that is not decomposed further satisfies its specification. In the setting of risk assessment, basic techniques are used for showing that a risk model *R* is correct for a target of analysis *T*, when the two are not decomposed further. Because such techniques should be used when the size or complexity of the target is manageable without further decomposition, this means that traditional risk assessment methods, techniques and tools are applicable. Importantly, in contrast to formal verification, where the aim is to formally prove that a program satisfies its specification, such rigor cannot be expected in general for risk assessment. Instead, the aim should be to substantiate the correctness of the risk model by means of the available evidence and documentation.

4.4 Target Composition

In the following we introduce our operators for composition of the target of analysis. Target composition is the last part of the foundation for compositional risk assessment presented in this section.

Given the trace sets $H \in \mathbb{P}(\mathbb{H}) \setminus \emptyset$ that we use to represent the target of analysis, we now define composition operators $\oplus$ to facilitate the composition $\oplus(H_1,...,H_n)$. In the following we introduce binary composition operators. However, by associative and commutative properties the operators can be applied also for several decompositions. We refer to the appendix for the formal definitions. Below we only explain intuitively what the operators mean. After the presentation of the composition operators we explain the risk graph semantics for these operators.

4.4.1 Composition Operators

Parallel composition: For trace sets H_1 and H_2, the parallel composition $H_1 \| H_2$ yields the set $H \subseteq \mathbb{H}$ in which each trace of H is an interleaving of one trace from H_1 and one trace from H_2. Parallel composition is associative and commutative.

Sequential composition: For trace sets H_1 and H_2, the sequential composition $H_1 ;H_2$ yields the set $H \subseteq \mathbb{H}$ in which each trace of H is the concatenation of one trace from H_1 and one trace from H_2. For traces $h_1 \in H_1$ and $h_2 \in H_2$ in which h_1 is infinite, the concatenation yields h_1. Sequential composition is associative.

Non-deterministic choice: For trace sets H_1 and H_2, the non-deterministic choice composition $H_1 \square H_2$ yields the union of the two sets. Non-deterministic choice is associative and commutative.

Features: When applicable, these standard kinds of composition may be useful also in the setting of risk assessment. However, such bisected decompositions are often not adequate for the way different parts of a system are assessed in practice. Instead, one specific risk assessment may be concerned with certain actors and work processes regarding certain aspects of the system, while another assessment may be concerned with a different set of actors, processes and aspects. Yet, the two assessments of different parts of the same system may overlap in many ways. For example, one risk assessment may address the service consumers of a web application and focus on the confidentiality of certain information, while another assessment of the same system may address the service provider and focus on integrity. Another example is the decomposition of a system according to different views [16], such as an information viewpoint, an enterprise viewpoint and a technology viewpoint. If risk assessments are conducted separately for the different views, specific compositional techniques are required for deducing the combined result.

In our compositional approach to risk assessment we use the notion of *feature* to refer to such parts or aspect of a system. Formally, given the trace set H representing the combined target of analysis, a feature F may be represented by a set of events $E \subseteq \mathbb{E}$. In that case, the separate risk assessment is with respect to the subset of H that consists of the traces in which any of the events in F occur. We also investigate a more general representation of features in which F is represented by a set of sub-traces of H. A trace h' is a sub-trace of h, denoted $h' \lhd h$, when the events in h' also occur in h, and in the same order. For example, the trace of elements <*b*, *f*, *g*> is a sub-trace of <*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*>, whereas <*b*, *f*, *i*> and <*b*, *g*, *f*> are not. In that case, the separate risk assessment is with respect to the subset of H such that for each trace in this subset, a trace exists in H such that the latter is a sub-trace of the former.

Whether a feature F is represented by a set of events or a set of sub-traces, we use $F \ltimes H$ to denote the target H with respect to the feature F. For separate features F_1 and F_2 with respect to the same target of analysis H, we will introduce compositional proof rules for the composition $F_1 \oplus F_2 \ltimes H$. We also envisage compositional proof rules for different features with respect to different targets, i.e. $F_1 \ltimes H_1 \oplus F_2 \ltimes H_2$.

4.4.2 Risk Graph Semantics for Composition Operators

Having introduced composition operators for the representation of the target, we need to define the semantics of risk graphs with respect to composition.

For parallel composition, sequential composition and non-deterministic choice this is straightforward. For each of these operators, the result of composing two sets of histories is a set of histories. Hence, the definition of the semantics is as explained in Section 4.2.

For a feature F, $F \ltimes H$ captures a subset of H. We could therefore choose to define the semantics directly for the resulting subset. However, in order to keep the information about the different features and the target systems they refer to, we need to represent them as a pair of events/sub-traces and histories in our formalism. In any case, for a feature $F \ltimes H$, a risk graph specification $F \ltimes H \vdash (V, R)$ are statements about the subset of H that is captured by F.

The reader is referred to the appendix for the full definition of the risk graph semantics for the target composition operators.

4.5 Conclusion and Ongoing Work

The objective of this chapter was to present the current status of our ongoing work on developing a compositional approach to security risk assessment. Based on this foundation, our goal is to develop a set of compositional proof rules for the sound composition of risk assessment results from different risk models.

By using binary composition operators$\oplus$, the compositional proof rules that we investigate in the ongoing work are of the following form.

$$\frac{H_1 \vdash (V_1, R_1) \qquad H_2 \vdash (V_2, R_2) \qquad [\mathit{additional\ conditions}]}{H_1 \oplus H_2 \vdash (V_1 \cup V_2, R_2 \cup R_2)}$$

In this case the decomposed risk model is combined by taking the union of the respective sets of risk graph vertices and relations. While this is sound under some conditions, we may also have rules in which we can deduce the correctness of only certain fragments of the risk model.

The underlying formalism presented in this section and in the appendix is rather technical. The purpose of the formal foundation, however, is to serve as an important tool for us as method developers to ensure rigor and to prove the soundness of the approach. End-users, i.e. risk analysts, should only need to apply the rules we develop directly on the risk models they create. What the risk analysts need are therefore methods and guidelines for how to apply the rules to enable a compositional risk analysis of large scale systems.

In order to ensure the applicability and usefulness of the approach we are in the ongoing work also developing realistic use case scenarios to evaluate and validate the approach.

5 Reusability of Risk Assessment Artifacts

In this section, specific concepts for component based risk analysis and the composition of reusable risk analysis artifacts are presented. Section 5.1 summarizes an extension for the CORAS risk assessment method to support formal composition of risk analysis artifacts. Section 5.2 deals with the need for some common standards to make risk analysis artifacts really reusable and shows how using powerful but complex standard formats can be made practically applicable.

5.1 Composition Techniques

The CORAS method itself does not offer much support for component based and compositional risk analysis. In [32] “Dependent CORAS Diagrams” are suggested to deal with dependencies of different components. These diagrams are appropriate to hide some complexity from the “context scenario” and for dealing with assumptions about the environment, which could then be replaced with risk analysis results about the environment. They are no solution for compositionality of risk analysis artifacts in general.

In [31], an extension of the CORAS method is suggested which is designed exactly to deal with the component based compositional risk analysis of complex systems. Here the most important results of that paper will be presented in a way which is more formal, precise and consistent with the abstract thoughts about compositionality shown in the previous sections. Some further extensions are suggested.

Let *S* be the complex system that should be analyzed which consists of different Components $\{C_1, \ldots, C_Z\}$.

5.1.1 Threat Interfaces

A *threat interface* represents both, an individual component and the risk analysis results for that component. It describes how an individual component could be influenced by other components and how it could itself affect the security of other components, but it hides internal details to reduce complexity.

To generate a *threat interface* T_X for some component C_X, the steps one to six of the conventional CORAS method are performed for C_X. This results in a *threat diagram* for C_X containing vulnerabilities, threat scenarios, unwanted incidents, arrows between those elements modeling relation paths and likelihood values.

The *threat interface* T_X is a quadruple consisting of a name N_X which should describe the component C_X and three lists V_X, U_X, D_X. The first list V_X contains the vulnerabilities identified for C_X that are exposed to other components together with the likelihood values of these vulnerabilities if any have been assigned. The second list U_X contains the unwanted incidents identified for C_X which might be a threat for other components together with the likelihood values of these incidents if any have been assigned. The third list D_X contains a directed relation with a likelihood value for each path that can be identified in the *threat diagram* for C_X having a vulnerability from V_X as starting point and an unwanted incident from U_X as end point. Such a path might contain multiple arrows and other elements like threat scenarios in the *threat diagram* and each single element within a path might have an assigned likelihood value, but it will be represented by a single relation in the *threat interface* having a single likelihood value. Hence, a single likelihood value for the entire path has to be calculated, i.e. a multiplication of the probability values if *Strict CORAS* (described in section 5.2) is used. All the internal details of the *threat diagram* for C_X are hidden in the threat interface T_X.

$T_X = (N_X, V_X, U_X, D_X)$

Threat interface T_X for component C_X has a graphic representation as a box with N_X as its title label, the elements of V_X on the left hand side and the elements of U_X on the right hand side using the same symbols for vulnerabilities, unwanted incidents and the same notation for their likelihood values as in a conventional CORAS *threat diagram*. Arrows with dashed lines represent the directed relations of D_X. Their likelihood values are written right under the arrows.

5.1.2 Threat Composition Diagrams

In order to do a compositional risk analysis of the complex system *S*, for *S* and for each of its base components C_1,… ,C_Z a separate *threat interface* T_S,T_1, … , T_Z has to be generated. The next step is to model how the components C_1,… ,C_Z are composed to *S* using their *threat interface*T_1, … , T_Z and T_S. Formally, this is defining an operator σ*Conditions* over the interfaces that represents the composition of the components. However, to be consistent with the rest of the CORAS method, it is desirable to use an intuitive graphic modeling concept for this step. Therefore, the *threat composition diagram* is introduced:

The *threat composition diagram* consists of two layers. The *component layer* contains information about how the base components themselves are combined to a complex component. The second layer contains information about the vulnerabilities and unwanted incidents identified for each individual component and about how these could affect one another. That layer is called the *directed graph of consequences*.

The relations between the entire components are modeled on the *component layer* as relations between the *threat interface* boxes using arrows with dotted lines. If a simple arrow is not enough to make the relation understandable, *description boxes* may be used to informally explain relations.

While each *threat interfaces* T_X itself becomes a node in the graph of the *component layer* in a *threat composition diagram*, its vulnerabilities listed in V_X and its unwanted incidents listed in U_X become nodes in the *directed graph of consequences*. Notice that this is a more detailed model than the typical risk graph having only one kind of vertices *V* (i.e. only incidents).

The internal relations of each T_X listed in D_X become edges in the *directed graph of consequences*. Additionally, if an unwanted incident of some *threat interface* could affect a vulnerability of another *threat interface*, that *trigger relation* is modeled as another edge in the *directed graph of consequences*. In the *threat composition diagrams*, the graphic representation for such a *trigger relation* is an arrow having a continuous line. Let R_S be the list containing all these trigger relations identified for the complete system *S*. The edges in a simple *directed graph of consequences* are $R_S \cap D_1 \cap ... \cap D_Z \cap D_S$.

Gates can be used to express complex *trigger relations*: For example, multiple unwanted incidents might be required to actually affect a specific vulnerability. Most common are simple binary operators (e.g. AND, OR), but also more complex gates like *voting gates* may be used. Formally, R_S can contain complex *trigger relation* elements with *gate function* specifications. Each element of R_S consists of a set of unwanted incidents containing at least one element, a *gate function* defining how these incidents are combined and exactly one vulnerability. In the *threat composition diagram*, a *gate* is graphically represented by a small square with a label representing its combinatory *gate function*. The entire *trigger relation* containing a *gate* is modeled by placing the *gate* box over the border of the vulnerability it might affect and by drawing a continuous lined arrow from each unwanted incident to the *gate* box. The *gate* box may contain multiple separate named input sections on the left hand side if the *gate function* is not commutative. Then arrows from unwanted incidents have to lead to the correct input section.

The *component layer* and the *directed graph of consequences layer* presented so far are only a description of the composition. They are the model of the operator σ*Conditions*(T_1, … , T_Z, T_S).

Once they have been specified, mathematical formulas can be applied to make compositional conclusions about the likelihood of incidents that can be triggered by base components. That is applying a second operator σ*Conclusions* over the conditions. The conclusions are noted directly within the *directed graph of consequences layer as the likelihoods for the derivable incidents.*

In the following, the most basic conditions and conclusions are shown for likelihoods expressed as probability values according to the Kolmogorov axioms [29]. As described in [30], for calculating probability values, the following notations and equations/formulas will be used here: The probability of some incident *X* is noted as P(*X*). The conditional probability for incident *X* given that it is known that incident *Y* occurs is noted as P(*X* |*Y*). The probability P(*X* ∩ *Y*) that both incidents *X* and *Y* occur can be calculated with P(*Y*) and P(*X* |*Y*):

$$\mathrm{P}(X \cap Y) = \mathrm{P}(X|Y) * \mathrm{P}(Y) \qquad (1)$$

The probability $\mathrm{P}(X \cup Y)$ that at least one of two incidents X, Y occurs is:

$$\mathrm{P}(X \cup Y) = \mathrm{P}(X) + \mathrm{P}(Y) - \mathrm{P}(X \cap Y) \tag{2}$$

If *X* and *Y* are statistically independent (i.e. P(*X* | *Y*) = P(*X*) and P(*Y* | *X*) = P(*Y*)), then:

$$P(X \cap Y) = P(X) * P(Y) \tag{3}$$

$$\mathrm{P}(X \cup Y) = \mathrm{P}(X) + \mathrm{P}(Y) - P(X) * P(Y) \tag{4}$$

The first basic set of conditions is: Let *S* be a system which has two base components *C₁* and *C₂*. *S* uses *C₁* and *C₂* in a redundant way.

Let $T_S = \left(N_S, V_S = \{W_S\}, U_S = \left\{\left(I_S, P_S = P(I_S)\right)\right\}, D_S = \{W_S \rightarrow^K I_S\}\right)$ be the threat interface generated for *S*.

Let $T_1 = \left(N_1, V_1 = \{W_1\}, U_1 = \left\{\left(I_1, P_1 = P(I_1)\right)\right\}, D_1 = \{W_1 \rightarrow^M I_1\}\right)$ be the threat interface generated for *C₁*.

Let $T_2 = \left(N_2, V_2 = \{W_2\}, U_2 = \left\{\left(I_2, P_2 = P(I_2)\right)\right\}, D_2 = \{W_2 \rightarrow^L I_2\}\right)$ be the threat interface generated for *C₂*.

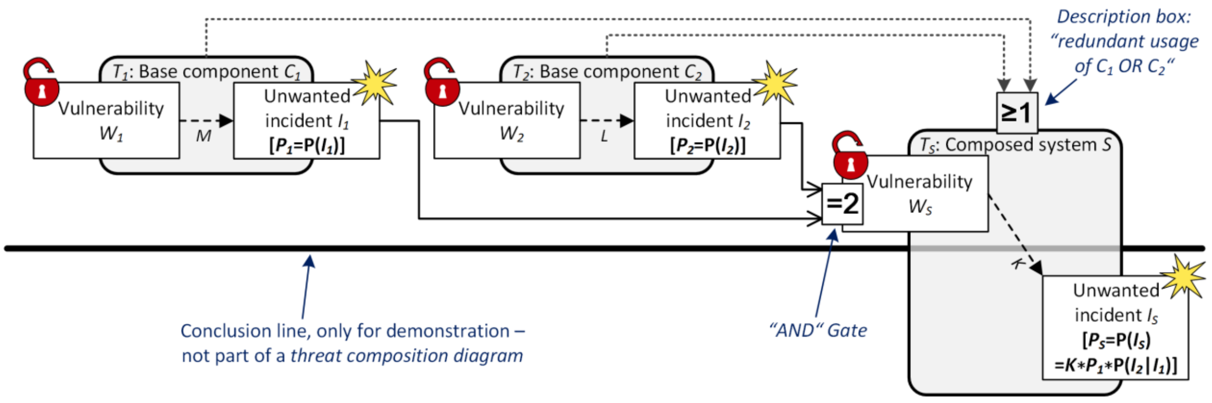
According to the description of *S*, *E_S* will be: $E_S = \left\{(T_1, T_2) \xrightarrow{\text{"redundant usage of } C_1 OR\ C_2\text{"}} T_S\right\}$

Let there be only a single identifiable *trigger relation* for the entire system *S*: Incident *I_S* will be triggered if and only if both base incidents *I₁* and *I₂* occur. Hence, $R_S = \{I_1 \cap I_2 \rightarrow W_S\}$.

Using the conditions and applying formula (1), it becomes possible to conclude:

$$\begin{array}{ll} \sigma Conditions: & T_1 \vdash (I_1, P_1) \quad T_2 \vdash (I_2, P_2) \quad R_S = \{I_1 \cap I_2 \rightarrow W_S\} T_S \vdash D_S = \{W_S \rightarrow^K I_S\} \\ \hline \sigma Conclusions: & P_S = \mathrm{P}(I_S) = K * \mathrm{P}(I_1) * \mathrm{P}(I_2|I_1) = K * P_1 * \mathrm{P}(I_2|I_1) \end{array}$$

Using the graphical representation of the *threat composition diagram*, this looks like the diagram shown in Figure 5.



Description box: "redundant usage of C₁ OR C₂"
T₁: Base component C₁
Vulnerability W₁
M
Unwanted incident I₁ [P₁=P(I₁)]
T₂: Base component C₂
Vulnerability W₂
L
Unwanted incident I₂ [P₂=P(I₂)]
≥1
T_S: Composed system S
=2
Vulnerability W_S
K
Unwanted incident I_S [P_S=P(I_S) =K*P₁*P(I₂|I₁)]
Conclusion line, only for demonstration – not part of a threat composition diagram
"AND" Gate

Figure 5 – Threat composition diagram with AND gate

For example, if *I₁* and *I₂* would be statistically independent from one another (i.e. $P(I_2|I_1) = P_2$) then it would be possible to apply formula (3) and to calculate $P_S = K * P_1 * P_2$ if just *K*, *P₁* and *P₂* were known. It is shown in [] how to analyze for statistical dependency using the compositional risk analysis method described here.

The second basic set of conditions is: Let *F* be a system which has the same two base components *C₁* and *C₂* but *F* uses *C₁* and *C₂* in a parallel way.

Let $T_F = \left(N_F, V_F = \{W_F\}, U_F = \left\{\left(I_F, P_F = P(I_F)\right)\right\}, D_F = \{W_F \rightarrow^H I_F\}\right)$ be the threat interface generated for *F*.

Let *C*$_1$ and *C*$_2$ be as defined above.

According to the description of *F*, *E*$_F$ will be: $E_F = \left\{(T_1, T_2) \xrightarrow{\text{"}parallel\ usage\ of\ C_1 AND\ C_2\text{"}} T_F\right\}$

Let there be only a single identifiable *trigger relation* for the entire system *F*: Incident *I*$_F$ will be triggered if and only if both base incidents *I*$_1$ and *I*$_2$ occur. Hence, $R_F = \{I_1 \cap I_2 \rightarrow W_F\}$.

Using the conditions and applying formula (2) and (1), it becomes possible to conclude:

$$\frac{\sigma Conditions: \quad T_1 \vdash (I_1, P_1) \quad T_2 \vdash (I_2, P_2) \quad R_F = \{I_1 \cup I_2 \rightarrow W_F\} T_F \vdash D_F = \{W_F \rightarrow^H I_F\}}{\sigma Conclusions: \quad P_F = \mathrm{P}(I_F) = H * \mathrm{P}(I_1) + \mathrm{P}(I_2) - \mathrm{P}(I_1) * \mathrm{P}(I_2|I_1) = H * P_1 + P_2 - \mathrm{P}(I_1 \cap I_2)}$$

Using the graphical representation of the *threat composition diagram*, , this looks like the diagram shown in Figure 6.

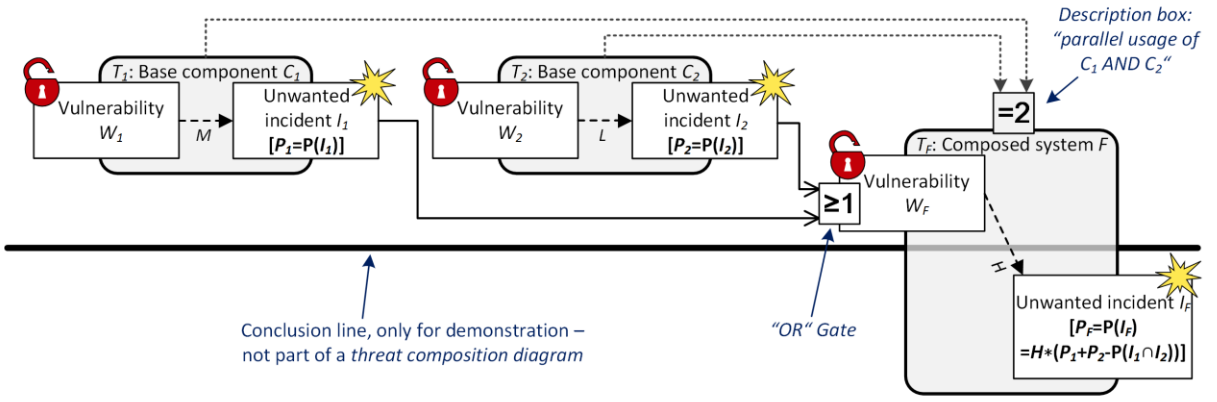

Description box: "parallel usage of C1 AND C2"
T1: Base component C1
Vulnerability W1
M
Unwanted incident I1 [P1=P(I1)]
T2: Base component C2
Vulnerability W2
L
Unwanted incident I2 [P2=P(I2)]
=2
TF: Composed system F
Vulnerability WF
≥1
H
Unwanted incident IF [PF=P(IF) =H*(P1+P2-P(I1∩I2))]
Conclusion line, only for demonstration – not part of a threat composition diagram
"OR" Gate

Figure 6 – Threat composition diagram with OR gate

For example, if *I*$_1$ and *I*$_2$ would be statistically independent from one another (i.e. $P(I_2|I_1) = P_2$) then it would be possible to apply formula (3) and to calculate $P_S = H * (P_1 + P_2 - P_1 * P_2)$ if just *H*, *P*$_1$ and *P*$_2$ were known.

The *threat composition diagram* can be used to model statistical dependencies and it can deal with the different dependencies multiple triggers for each incident might have by using table-like *dependency sets* to keep them separate. The reader is referred to [31] for the details.

5.1.3 Risk Comparison Diagrams

It is possible to create a conventional CORAS risk diagram from a *threat composition diagram*. However, a risk diagram cannot represent any component specific information. It might be beneficially to create a diagram that keeps some information about the components in order to make components or complex combinations of components comparable in terms of risks. It should be possible to choose the architecture with the fewest risks and to identify the most critical components at a glance. That would allow focusing test and treatment efforts on these most crucial elements. Therefore, the *risk comparison diagram* was introduced in [31]. The idea is to represent each component by a risk table that represents the different risk values in its rows. Relations from identified risks to assets are then modeled with lines passing through their components risk table exactly in the row representing the risk value. Different configurations of components are modeled as separate components with their own risk tables.

5.2 *Strict CORAS* for Better Reusability

In step four of the CORAS method the scales for expressing likelihoods, consequences and the functions to calculate risk values are defined by those who do the risk analysis. This flexibility is great but the freedom for the analysts makes it eventually difficult to reuse results of the risk analysis for different components if they use deviating scales and risk functions. Additionally, tools that should support the analyst e.g. by calculating probability values in the *threat composition diagram* would have to deal with an infinite number of different constructs.

The idea of *Strict CORAS* is to support both, the flexibility to define custom formats and scales but also to make sure that results are compatible and can be understood by standard tools. Using identical constructs for the analysis does not necessarily mean that the results from different persons and teams will look similar and understandable for each other. To further improve the reusability of risk analysis artifacts, *Strict CORAS* also contains design guidelines for the diagrams that are created. *Strict CORAS* is a standard suggestion and currently under development.

5.2.1 Standard Scales and Formats for the Likelihood

The simplest solution to avoid compatibility issues would be to provide default standard scales and formats that have to be used by anyone. However, it will for example be very difficult to find a single format for expressing likelihoods that is perfectly suitable for each possible scenario. If it is mighty and flexible enough to express all important aspects it will probably be too complicated for the analysis of simple systems.

It is eventually a much better idea to use a powerful standard format to express likelihoods as an internal base, but to allow the users to use more intuitive simpler formats if they need less features or less precision. However, to conform to *Strict CORAS*, it must be possible to convert likelihoods, consequences and risk values noted in less complex formats to the standard format.

So what is required to express the likelihood of some single incident accurately? Probability values according to the Kolmogorov axioms [29] are a very base notation for likelihoods. Noting a likelihood of 0.5 in that format means there is a 50% chance that it will happen. For coin tossing for example, this is sufficient.

However, for many real world incidents, time is the most important factor that additionally needs to be taken into careful consideration when analyzing and describing likelihoods. For example, radioactive decay cannot be described accurately by using just probability values since any instable atom will decay someone. So that value would always be one. Typically, half-life values expressed in some time format are used to describe such a process like radioactive decay. It is possible to interpret such likelihood values as a probability value of 0.5 per time period.

Reality can be even more complex: the probability per time period might change over time. Many systems get less robust over time and failures become more likely, for example. Therefore, it is absolutely necessary to be able to describe the likelihood as a function over the time.

Strict CORAS uses as an internal base format a probability function $P(T_1, T_2, T_S, L_X)$ taking a start point of time T_1, an end point of time T_2 and two parameters T_S and L_X as arguments to express the likelihood for the occurrence of some incident in the time between T_1 and T_2 as a probability value according to the Kolmogorov axioms. In *Strict CORAS* time values are generally expressed in seconds since the start point of time of the International Atomic Time (TAI). The parameters T_S and L_X may be used to define a likelihood function relative to a system dependent point of time. T_S specifies the moment when the system gets operational for the first time. L_X is used to specify the point of time when evaluation of the function should start or end (i.e. the occurrence of some base incident). L_X is intended especially for dependent incidents that can be triggered by other incidents with a certain probability. Note that incidents might occur and end multiple times. Therefore, L_X is a list containing eventually multiple start points of time, each optionally followed by an end point of time.

The *Strict CORAS* file format for actually specifying the function $P(T_1, T_2, T_S, L_X)$ is going to be a XML format like OpenMath / MathML [33].

5.2.2 Convertors and Examples for *Strict CORAS* Likelihood Notation

Defining a probability function $P(T_1, T_2, T_S, L_X)$ might be difficult. Tools should hide this complexity and allow the user to use more intuitive formats and notations. To be compatible with *Strict CORAS*, for each custom format a convertor to the *Strict CORAS* format has to be provided. The generation of *Strict CORAS* notations can then be done automatically.

As an example, a conversion from the simple probability value per time period format to the *Strict CORAS* format is shown here in detail. In the simple custom format, the likelihood for incident I is just a pair of a probability value Φ and a time period Δ. Then in *Strict CORAS* the likelihood that the incident I occurs between T_1 and T_2 can be expressed as:

$$\mathrm{P}(T_1, T_2, T_S, L_X) = 1 - (1 - \Phi)^{\frac{T_2 - T_1}{\Delta}} \tag{5}$$

Another risk analyst might prefer to use frequencies instead of probability values per time period to express likelihoods. Frequencies are somehow intuitive and very appropriate for some incidents. But additional information is required in order to understand what a frequency of 1 per time period Δ really means. It could mean that in each time period Δ there is exactly one occurrence of incident *I*. However, it could also mean that only in the long term on average, there will be approximately one occurrence of incident *I* for each period Δ, but there might be time periods of length Δ having no incidents *I* and there might be time periods of length Δ in which incident *I* occurs several times. There might be additional values specifying how variant the occurrence of the incident *I* is.

The *Strict CORAS* likelihood format can express such differences precisely, so these should be clarified before starting the conversion. If for each time period Δ there is exactly one occurrence of incident *I* and if incident *I* occurs for the first time Ω seconds after the point of time T_S when the system becomes operational then the following function can be used:

$$P(T_1, T_2, T_S, L_X) = \begin{cases} 1 & \text{if } \exists\, \mathrm{n} \in \mathbb{N} \mid T_2 - T_1 > T_2 - (T_S + \Omega + \Delta * n) \geq 0 \\ 0 & else \end{cases} \tag{6}$$

If the frequency is less precisely known and if there might be time periods of length Δ having no incidents *I* and there might as well be time periods of length Δ in which incident *I* occurs several times, then equation (5) with a chosen value Φ close to but slightly below one would be a good approximation using the *Strict CORAS* likelihood format.

5.2.3 Composition with *strict CORAS* Probability Functions: Risk Simulation and Risk Testing

One of the most important aspects in compositional risk analysis is to calculate likelihood values for triggered incidents. For simple constant probability values it is relatively easy to do such calculations according to the rules of probability theory. Formal methods can be used to get exact symbolic solutions.

With the probability functions defined in *Strict CORAS*, it is much more difficult to do precise calculations and to find symbolic solutions for the probability of triggered incidents: The likelihoods are in general considered to be time dependent, they might change dynamically. Probability values can only be calculated for specific time periods. The result of the probability functions might be state dependent. P(T_1, T_2, T_S, L_X) might use the parameters T_S and L_X which are determined by the previous state of the system. Eventually, it is not possible to calculate probability values for the time period from T_1 till T_2 without calculating probability values for the time period between T_S andT_1.

The idea for calculating likelihood values for triggered incidents in compositional risk analysis with likelihoods expressed as *Strict CORAS* probability functions is to use Monte Carlo simulations. Monte Carlo simulations are widely used to analyze complex systems and especially for risk aggregation [27][28]. Let T_E be the point of time for which likelihood values should be evaluated. Assuming that the risk analysts have already modeled the Threat Composition Diagram, the utilization of Monte Carlo simulation for calculating likelihood values for dependent incidents involves basically two phases.

The first phase consists of a large number of simulations based on random values. Instead of calculating aggregated probability values, each simulation calculates for each incident whether it occurred or not based on random sample values. For all the simulations, a common evaluation time period δ for each iteration is chosen. δ should be small compared to the time period$(T_E - T_S)$ and δ must be a divisor of $(T_E - T_S)$. Let *N* be $(T_E - T_S) \div \delta$. Each simulation stores a state value S_I for each incident *I*. S_I is one if *I* occurred and null otherwise. All state values for the incidents are initially null. For each integer *i* starting at null and being smaller than *N*, a random value R_{Ii} for each incident *I* is generated. If R_{Ii} is smaller than $P(T_S + \delta * i,\ T_S + \delta * (i+1), T_S, L_X)$, then incident *I* occurred and the state value for *I* becomes one. Since some damage could be detected and repaired within a finite time, the state of S_I might change back to null if such a wanted incident modeled as a treatment occurs. Figure 7 shows an example. In the example, T_Y and T_Z are passed as an element of L_X to the probability function of incident I_D just like T_X and T_Z are passed to the probability function of I_R.

The second phase evaluates the results of all the simulations made in phase one. For each incident *I* the arithmetic mean about the resulting state values S_I indicating whether *I* occurred (value is 1) or *I* did not occur (value is 0). This mean value is then a good approximation for the aggregated probability $P_I(T_E)$ that incident *I* occurs at time T_E.

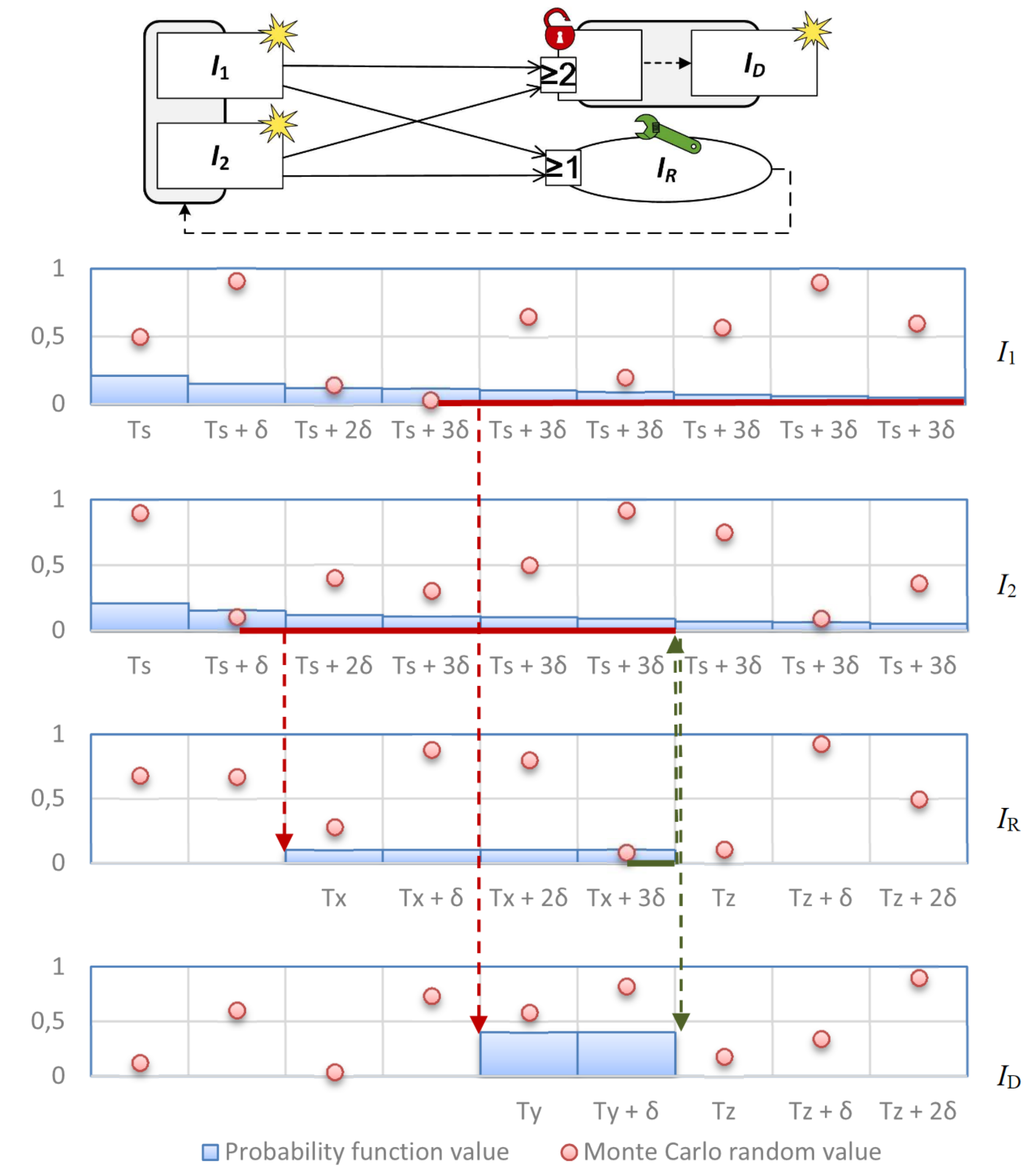


Figure 7 – *Threat Composition Diagram* with visualization of a Monte Carlo simulation for calculating whether incident I_D occurs or not.

Using *Strict CORAS* and compositional risk analysis should be flexible enough to express complex timing issues. Unwanted incidents might last only for a limited period of time. Some damage could be detected and repaired within a finite time. *Strict CORAS* should also allow modeling wanted incidents which could express the end of unwanted incidents. CORAS already has a concept for modeling treatments in treatment diagrams – these could eventually be used or a more general concept of wanted incidents could be introduced.

With Monte Carlo simulation, it becomes possible to calculate likelihood values even for complex systems with dynamically changing probabilities. It is easy to implement in tools. Solutions are not precise, but since the probability functions are based upon the analyst's expertise and usually incomplete information, the values are fuzzy, anyway.

Monte Carlo simulation for risk aggregation as described here is actually a kind of testing. However, it is not testing the complex system that is analyzed, but only a simplified model, i.e. the directed graph of consequences in a threat composition diagram. A model specifically created from a risk analysis perspective, containing the most crucial elements only. Evaluation of the probability functions might be much more efficient than testing the real system. Unfortunately, using probability functions created on guesses of the analysts might lead to wrong results. Hence in Monte Carlo simulation for compositional risk analysis, it might be a good idea to replace at least some most crucial or most uncertain assessed parts of the model with the real system and to do a security risk testing for these components while the rest of the system is only simulated.

We are currently investigating the full potential of Monte Carlo simulation combined with testing components of the real system and how this could be implemented in a tool.

5.2.4 Further work: Other Formats, Functions, Design Guidelines and a Mapping of Test Patterns to *Strict CORAS* Artifacts

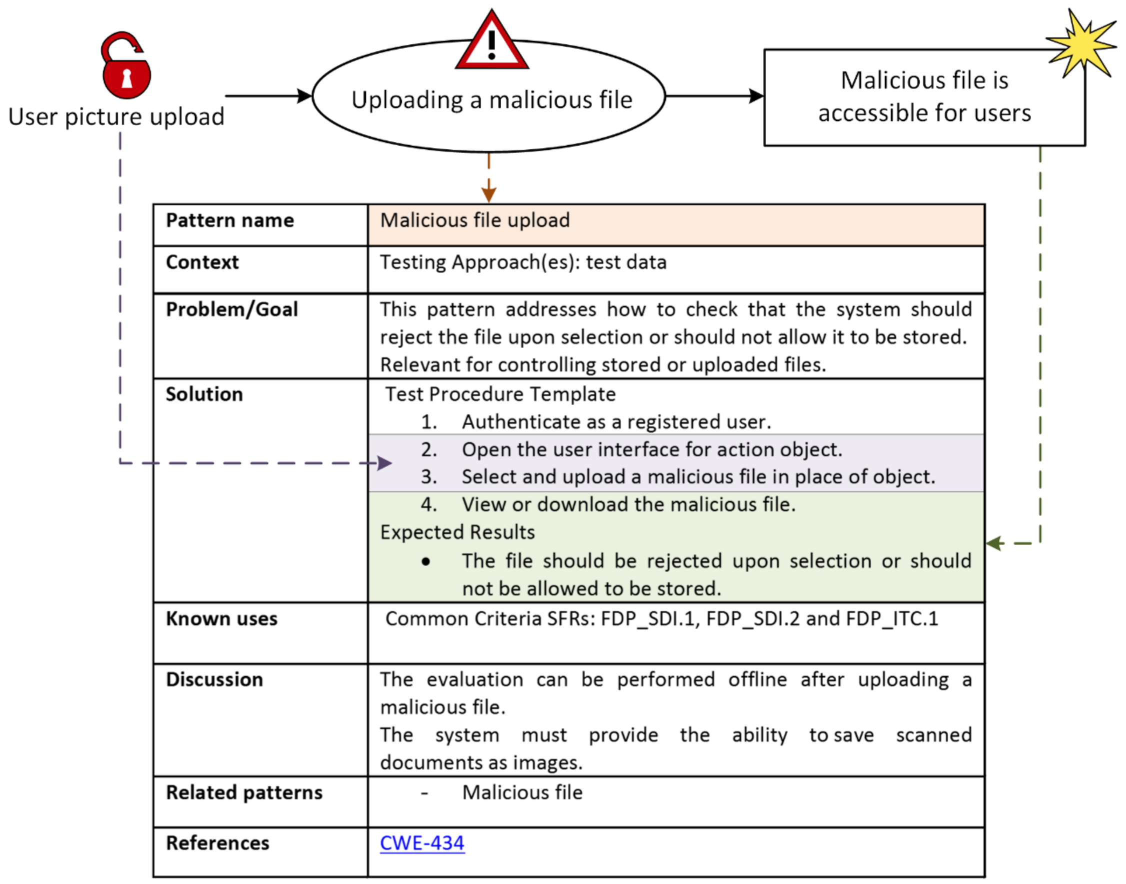
Strict CORAS will also contain a standard base format for the consequence values that supports dynamically changing consequences. Since both the likelihood format and the consequence format of *Strict CORAS* are going to be time dependent, the default risk function to calculate risk values will be time dependent, too.

Besides those standard formats and functions, *Strict CORAS* is going to contain also some design guidelines for the analysts. These are intended to make the reuse and integration of risk analysis artifacts produced by other risk analysts easier. The design guidelines might also be helpful for the interoperation between risk analysis and security testing.

Here some design guidelines for *Strict CORAS* threat diagrams are presented as an example: In a threat diagram, any relation starting from a vulnerability must lead to a thread scenario and for each vulnerability there must be at least one of these relations. Any relation starting from a threat scenario must lead to an unwanted incident and for each vulnerability there must be at least one of these relations. For the threat scenarios in a threat diagram designed according to these simple rules it is straight forward to create a threat interface or to identify and to apply an appropriate security test pattern.

There are good reasons why test patterns should be used if risk analysis is combined with security testing: It can be challenging to create effective test cases and to create appropriate metrics that allow sound conclusions for the likelihood values in the threat diagrams. If security testing for some real components should be used while running Monte Carlo simulations, then a systematic process not necessarily producing high manual effort is desirable for test case generation. Instead of reinventing the wheel each and every time, it makes sense to create and to use a catalogue of test patterns[26].Security test pattern do typically consist at least of a name, a context, a problem and a solution description.

If *Strict CORAS* is used and the design guidelines described above are applied, then the threat scenario is a direct counterpart to the problem description of a test pattern. The solution description of a test pattern will typically contain some generic input parameters and some output results. The input parameters should be mapped to the vulnerabilities that have relations leading to the threat scenario. Vulnerabilities will be used as the input ports to pass test values to the system under test. The output results can be mapped to the unwanted incidents that might be triggered if the threat scenario becomes real. Figure 8 shows an example mapping.


User picture upload
Uploading a malicious file
Malicious file is accessible for users

Pattern name	Malicious file upload
Context	Testing Approach(es): test data
Problem/Goal	This pattern addresses how to check that the system should reject the file upon selection or should not allow it to be stored. Relevant for controlling stored or uploaded files.
Solution	Test Procedure Template 1. Authenticate as a registered user. 2. Open the user interface for action object. 3. Select and upload a malicious file in place of object. 4. View or download the malicious file. Expected Results • The file should be rejected upon selection or should not be allowed to be stored.
Known uses	Common Criteria SFRs: FDP_SDI.1, FDP_SDI.2 and FDP_ITC.1
Discussion	The evaluation can be performed offline after uploading a malicious file. The system must provide the ability to save scanned documents as images.
Related patterns	- Malicious file
References	CWE-434

Figure 8 – Mapping threat diagram artifacts to test pattern

Within the RASEN project we plan to develop a tool that supports the Strict CORAS standards and which can be used to do Monte Carlo simulations in combination with pattern based security tests for selected real components to improve the risk analysis.

6 Complement the Risk Picture Using Test Results

6.1 Introduction

As previously stated in RASEN deliverable D3.1.1, the possible influence of security testing activities on risk assessment includes the following:

- **Risk identification**. Application security testing helps to discover unknown vulnerabilities, thereby contributing to the security risk identification.
- **Risk analysis**. The potential impact of detected vulnerabilities depends often of multiple conditions regarding product usage. Accurate security testing helps to avoid false positive (incorrect detection of a vulnerability) and then to better estimate likelihood and consequences of threat scenarios.
- **Risk evaluation**. Detection of vulnerabilities through application security testing helps to identify known vulnerabilities. For example, in web applications, this may refer to well establish vulnerability identification database (such as CVE) and then to more precisely measure the risk level using for example the Common Vulnerability Scoring System (CVSS)[34].
- **Risk treatment**. Depending of risk assessment phases, remediation actions may be decided and implemented. Then, security testing is used to confirm the mitigation of identified risks.

In this section, we present several approaches that are investigated in RASEN to support risk assessment based on security test results.

6.2 Test-Based Risk Identification, Estimation and Treatment

The testing process and the corresponding generated testing artifacts can actively contribute to identify, estimate and verify security risks of the tested application. For each of these risk management activities, dedicated testing techniques indeed enable to produce the following data:

6.2.1 Techniques for Test-Based Risk Identification

Complementing the risk identification can be performed by managing a risk traceability matrix between vulnerabilities and generated test cases. This traceability matrix links each test case to the vulnerabilities (identified and classified regarding existing databases, such as CVE) that it tests.

The risk identification can also be completed using an organic traceability matrix between application parts (structural or component feature) and the generated test cases. In this matrix, each test case is linked to one or more parts of the application under test (regarding functional and/or architectural description of the application), which are covered by its corresponding purpose.

6.2.2 Techniques for Test-Based Risk Estimation

Risk estimation can be completed using the testing context and the test data, such as inputs, outputs and results of the executed test cases. On the one hand, to make these data relevant for estimation, it needs techniques to assign test verdict avoiding, as much as possible, false positive results. On the other hand, we need to be able to determine the significance of the detected vulnerabilities identified and classified regarding existing approaches (such as CVSS framework).

6.2.3 Techniques for Test-Based Risk Treatment Verification

In order to help risk treatment activity, testing approach can offer the capacity to record and re-execute previously generated test cases to ensure non regression. This tooling can thus be used to confirm the mitigation of detected risks.

The next section introduces the proposed testing RASEN approach, which develops novel techniques to produce results aiming at complementing the risk picture.

6.3 RASEN Test Data for Complementing Risk Picture

To address the risk picture issues, Figure 9 depicts the RASEN overall testing process, which implements the techniques introduced in the previous section to complement the risk picture.

This RASEN process takes as input a set of targeted vulnerabilities, identified by the risk model, and the behavioral description of the application under test, specified by an environmental test model. From these inputs, the RASEN test generation tool, mainly based on Smartesting model-based testing tool and Fraunhofer FOKUS fuzzing tool, enables to generate security test cases using techniques driven by the identified risk. The generated test cases take the form of sequences of abstract operation calls with the corresponding expected results (they are described at the abstraction level of the initial test model). These abstract test cases are then automatically concretized using a dedicated adaptation layer in order to provide executable test scripts and to automatically assign the verdict of the test (by comparing the expected results and the results observed during the execution of the tests).

This process supports the traceability of the targeted risk and vulnerability threats: this capability makes it possible to link each generated test case with the vulnerabilities it covers, and to know which test cases cover which vulnerabilities.

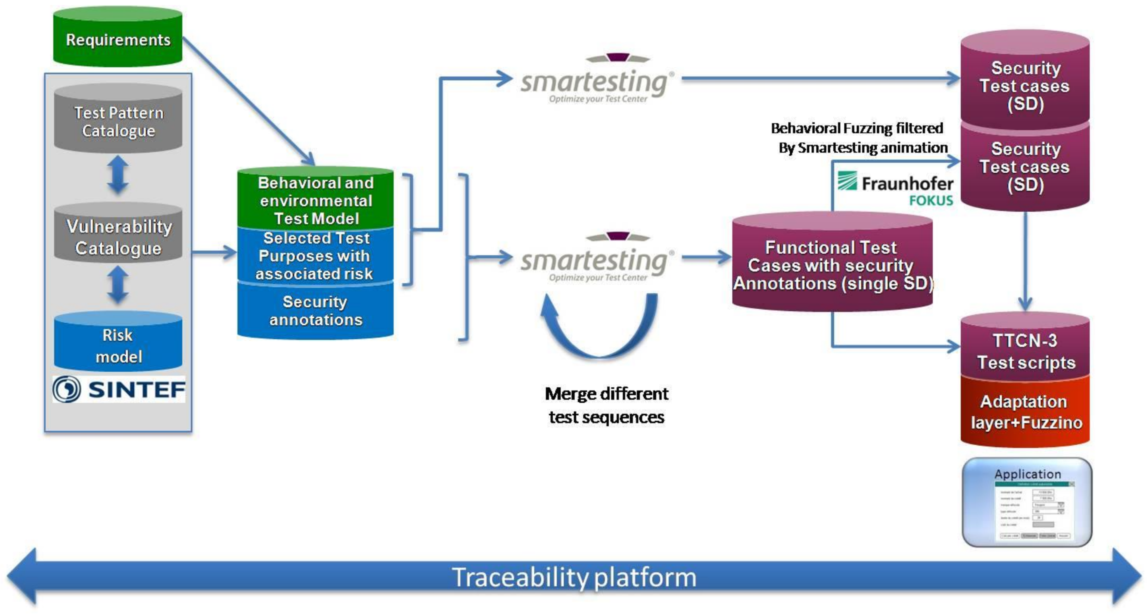

Requirements
Test Pattern Catalogue
Vulnerability Catalogue
Risk model
SINTEF
Behavioral and environmental Test Model
Selected Test Purposes with associated risk
Security annotations
smartesting Optimize your Test Center
smartesting Optimize your Test Center
Merge different test sequences
Behavioral Fuzzing filtered By Smartesting animation
Fraunhofer FOKUS
Functional Test Cases with security Annotations (single SD)
Security Test cases (SD)
Security Test cases (SD)
TTCN-3 Test scripts
Adaptation layer+Fuzzino
Application
Traceability platform

Figure 9 – Global testing process overview

During this process, a set of artifacts and information is produced. Figure 10 shows the testing process by focusing on the used (inputs) and produced (output) artifacts managed by each tool component during the testing process. The red arrows correspond to the data exported from the testing process to risk assessment tool, and thus they can be used to define / complete the risk picture of the tested application.

The exported test data for risk assessment are the following:

- Purpose of each test case trace in terms of targeted vulnerability (risk traceability matrix) and targeted part of the application under test (organic traceability matrix).
- Executed test case traces including all performed actions with potential filled data.
- Expected effects of each action of the test traces, from the behavioral model point of view.
- Observed effects of each action of the test traces, from the execution point of view.

- Test verdict in terms of identification of test success (discovered vulnerability) and failure (absence of vulnerability), and more generally feedback when something unusual or unexpected has occurred during the trace execution.

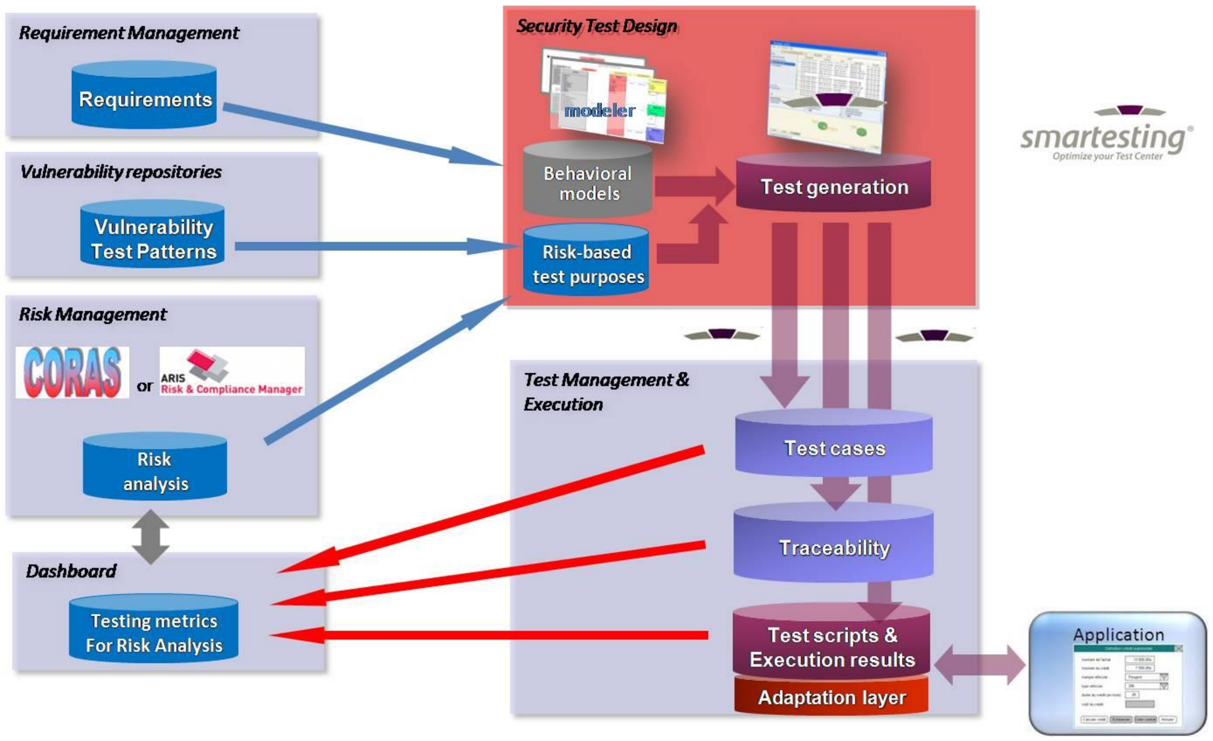


Figure 10 – Testing artifacts exchange

6.4 Summary

This section has introduced the techniques and related test data that can be used to complement the risk picture, and thus to improve the risk assessment activity. The testing approach, developed within the WP4 of the RASEN project, has been briefly showed to highlight what will be proposed by the testing RASEN tooling, in terms of techniques and generated data, to complement the risk picture.

7 Aggregating Test Results Using Security Indicators and Risk Metrics

In this section we address the challenge of how to aggregate low-level test results in order to use the results as input to the security risk assessment. This challenge is relevant for two of the WP3 tasks, namely the development of techniques and tools for test-based risk identification and assessment, and the development of methods for continuous risk assessment by means of test-based indicators. In both cases we need techniques for how to analyze the empirical data to support the risk assessment. The techniques should support the WP3 approach to complementing the risk picture using test results as described in Section 6, but also to empirical data gathered from security and risk monitoring. The former is referred to as active testing, whereas the latter is referred to as passive testing.

7.1 Method Overview

Figure 11 of RASEN deliverable D5.3.1 shows our generic method for test-based security risk assessment. On the one hand, before the risk identification starts, security testing should serve as a means for directing the risk identification by the identification of important vulnerabilities and potential incidents that should be investigated further by the risk analysts. On the other hand, after the risk assessment, security testing should serve as a means to validate and/or correct the risk assessment results. It is the latter use that is the topic of this chapter.

As depicted in Figure 11, our method follows an iterative process where testing can be conducted one or more times to support the risk assessment, or it can be a more continuous process where the risk assessment results are updated on a regular basis by means of monitoring. The risk assessment includes the three activities of risk identification, risk estimation and risk evaluation.

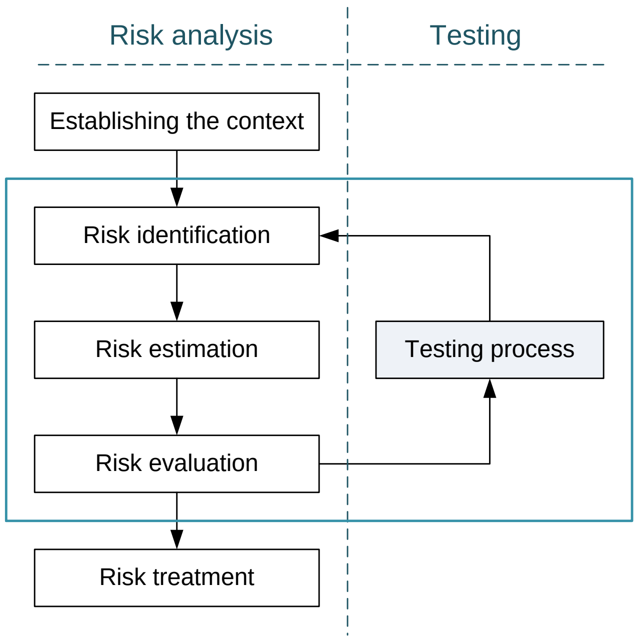


Figure 11 - Test-based risk assessment

7.2 Concepts and Definitions

The most central concepts of our approach and how they are related are depicted in Figure 12. The objective is to gather the empirical data that is needed to build or maintain the *security risk model* which, in principle, and be of any risk modeling language. However, the model must have the expressiveness to capture the information that is gathered from the testing. In the RASEN project we require that the risk model conforms to the conceptual models that we develop in the context of WP5. See deliverable D5.3.1.

Relative to the security testing, the risk models are a high-level representation of the vulnerabilities of a system and the incidents that may occur. As mentioned above and as shown in Figure 12, we will investigate the use of both *active security testing* and *passive security testing* (monitoring) to collect the empirical data that needs to be aggregated. As much as possible, the approach should be quantitative and provide explicit measures and indicator values that can be combined and aggregated. For the active security testing we aim to assess security properties by means of *security measurements*. A measurement provides a single-point-in-time view of a specific discrete factor [17]. A *security measurement* is in particular a value of an assessable security property of a system entity. For the passive security testing we aim to monitor the security of systems by the monitoring of *security indicators*. An indicator is something that provides a clue to a matter of larger significance or makes perceptible a trend or phenomenon that is not immediately detectable [7]. For example, an unexpected rise in the traffic load of a web server may indicate that a denial of service attack is in progress.

As illustrated in the figure, from the viewpoint of the risk assessment, the security measurements and the security indicators values are raw, objective data that has yet to be interpreted. Following [17], we use the term metric to refer to the interpretation of such data, i.e. the results of some analysis of the empirical data. More precisely, *security metrics* involve the application of a method of measurement to one or more entities of a system that possess an assessable security property to obtain a measured value. A security metric can, for example, be the quantification of the degree of freedom from the possibility of an attack [17]. Hence, the security metrics should be the results of the interpretation of a set of security measurements and a set of security indicator values.

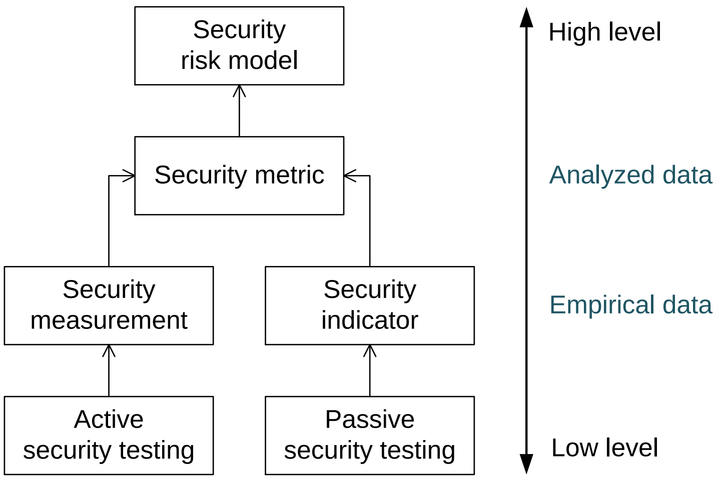


Figure 12 – Aggregating test results

In the setting of information security as defined by ISO 27001 [12] and information security risk management as defined by ISO 27005 [13], we will carefully consider the series on information security indicators by the European Telecommunications Standards Institute (ETSI) [6]. The purpose is to aid organizations in performing an objective assessment of their Information Security Management System (ISMS) and of the residual risk. This includes the assessment of the possible security incidents and vulnerabilities, which is a core objective of RASEN WP3. The ETSI security indicators are those that are considered common to all industry sectors, and are therefore also relevant for RASEN. The categories of security incidents are intrusions and external attacks, malfunctions and internal deviant behaviors, whereas the categories of vulnerabilities are behavior vulnerabilities, software vulnerabilities, configuration vulnerabilities and general security (technical or organizational) vulnerabilities. Currently in RASEN, we are mostly focusing on developing security vulnerability metrics for aggregating test results. Such a metric can be specified in terms of the probability that a vulnerability exists within the system under test, in combination with its degree of exploitability.

In the overview of Figure 12, aggregation of different kind of values can occur at several places and at different levels of abstraction. At the level of active or passive testing, for example, different test results can be combined for the purpose of making one security measure or security indicator, respectively.

Moreover, different measurements and indicators can in turned be combined in order to form one security metric. In RASEN WP3, where the main objective is to provide support for the security risk assessment, we will focus on the development of techniques and tools for how to take the security metrics as input and aggregate these in order to facilitate the risk assessment. An important part of the ongoing work is to refine the interface between the testing and the risk assessment in order to define exactly the data that shall be the output from and input to the different activities.

As mentioned above, we aim for a quantitative approach to aggregation of test results. In our development and reasoning about security metrics, we will carefully consider the mathematical notion of metric, how it is defined, and the conditions that a metric should satisfy. Basically, in mathematics [4], a metric on a set X (the metric space) is a function d that defines a distance between two elements in X. Properties that must be satisfied by d includes the following:

$d(x,y) = 0$ iff $x = y$ (*non-negativity*)

$d(x, y) = d(x, y)$ (*symmetry*)

$d(x, y) + d(y, z) \geq d(x, z)$ for any $x, y, z \in X$ (*sub-additivity*)

7.3 Example

Figure 13 illustrates how security measurements and security indicators can be linked to the risk assessment. In this case we have used a CORAS threat diagram with annotations to relate elements of the threat diagram to security indicators (SI) or security measurements (SM). How these values should be represented in practice and implemented in the RASEN tools is part of the ongoing work. The values should, however, be quantitative to represent, for example, a number detected events or a degree of fulfillment of a certain property.

Notice that in this example we have related the CORAS diagram directly to the measurements and indicators. This is for illustrative purposes, since we will use security metrics as a means to aggregate low level test results to derive the more high level data that is needed for the risk assessment. When the abstraction level of the risk assessment is too high to capture the link to specific test cases, we may also use high-level CORAS [32] to do a more detailed risk modeling of selected elements.

High-level CORAS comes with support for decomposing CORAS diagrams in a hierarchical manner, as illustrated in Figure 14. This diagram is a decomposition of the threat scenario *Malcode introduced by hacker via web application* from Figure 13. Using this decomposition we can specify at a more detailed level how this particular attack may be conducted. For such decompositions, also the metrics need to be decomposed. In the example, the security indicator *#detected attacks* from Figure 13 is decomposed into the three indicators *#detected XXS attacks*, *#detected SQL injections* and *#detected virus injections* in Figure 14. In addition, the security measure *probability of existence of vulnerability* has been added. In our development of techniques for specifying and aggregating security metrics, the high-level metric should be the result of an aggregation of the low-level metrics in the same way as a high-level CORAS diagram can be derived from its decompositions in High-level CORAS.

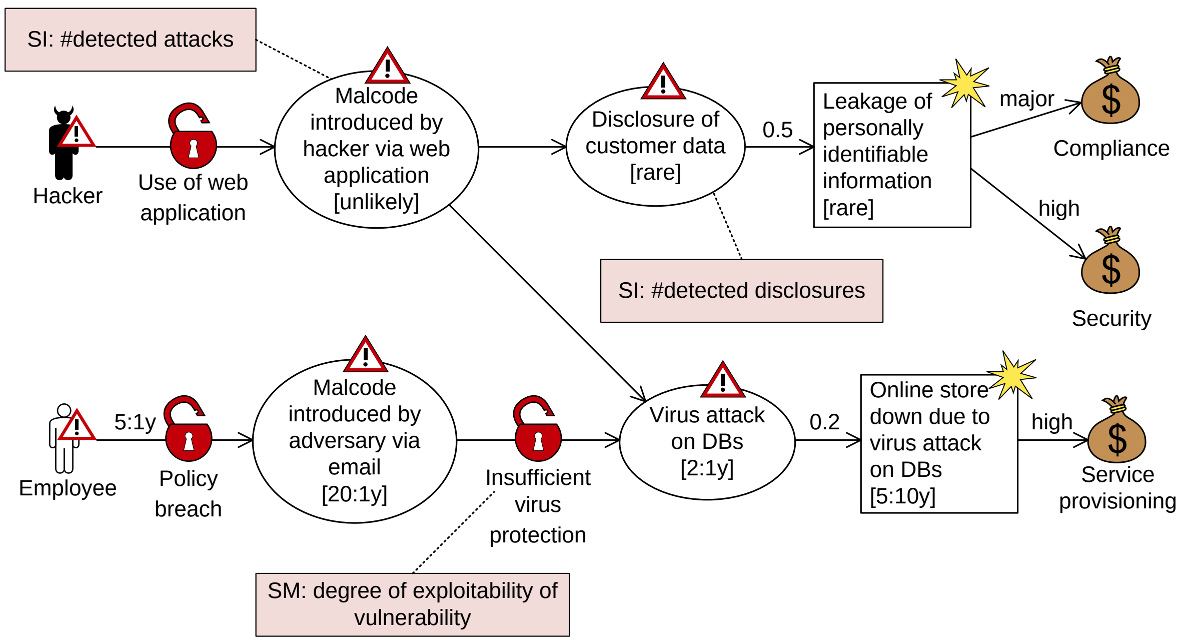

SI: #detected attacks
Hacker
Use of web application
Malcode introduced by hacker via web application [unlikely]
Disclosure of customer data [rare]
0.5
Leakage of personally identifiable information [rare]
major
Compliance
high
Security
SI: #detected disclosures
Employee
5:1y
Policy breach
Malcode introduced by adversary via email [20:1y]
Insufficient virus protection
Virus attack on DBs [2:1y]
0.2
Online store down due to virus attack on DBs [5:10y]
high
Service provisioning
SM: degree of exploitability of vulnerability

Figure 13 – CORAS diagram with security metrics

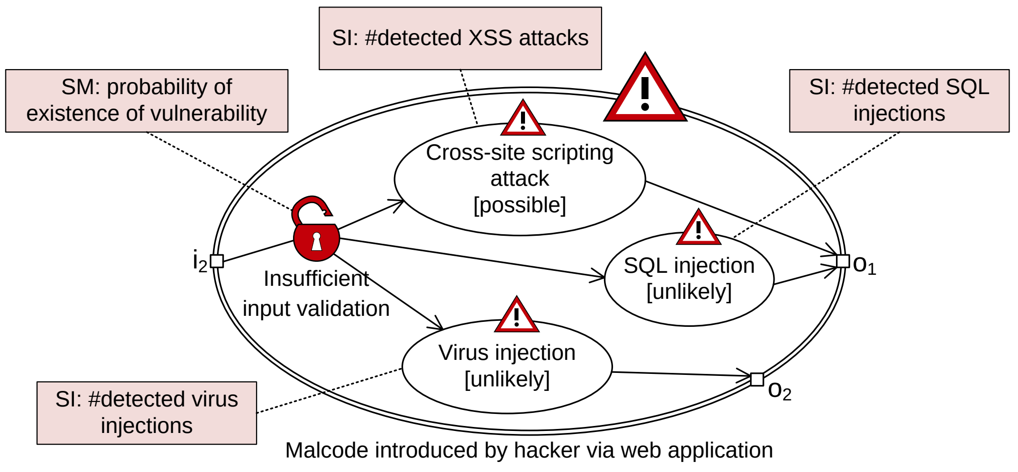

SI: #detected XSS attacks
SM: probability of existence of vulnerability
SI: #detected SQL injections
Cross-site scripting attack [possible]
i2
Insufficient input validation
SQL injection [unlikely]
o1
Virus injection [unlikely]
o2
SI: #detected virus injections
Malcode introduced by hacker via web application

Figure 14 – Decomposed CORAS diagram with security metrics

In our work on passive security testing and the use of security indicators to support the risk assessment we will build on our previous work on key indicator monitoring [19], as well as the CORAS risk monitor [18]. For the active security testing we will develop risk metric techniques for aggregating test results as described in Section 6.

7.4 Conclusion and Ongoing Work

In order to enable a systematic and sound method for conducting test-based security risk assessment, which is one of the main objectives of WP3, we need techniques for utilizing the empirical results of the testing. Referring to the overview in Figure 12 we are working on the following R&D tasks in WP3:

- The development of methods for the identification of security measurements and adequate test cases

- The development of methods for the identification of security indicators and adequate means for indicator monitoring
- The development of techniques for aggregating security measurements and security indicators values into security metrics
- The development of techniques for aggregating security metrics to derive the data and information that is needed for supporting the risk assessment and risk modeling

The WP3 techniques and tools for test-based risk assessment will be closely aligned with the techniques and tools for risk-based security testing. We refer to RASEN deliverable D5.4.1 for the description of the current status of the tool integration.

8 Conclusion

In this deliverable we have reported on our first results of the current WP3 tasks, namely compositional security risk assessment (task T3.1) and test-based security risk assessment (task T3.2).

In our work on developing a formal foundation for compositional risk assessment, we have defined a formal semantics for risk graphs and introduced operators for target composition. This foundation is an important basis for the development of compositional proof rules and for us to demonstrate the soundness of these rules. The advantage of using risk graphs is that they can be instantiated by several established risk modeling techniques. This means that our techniques and rules should be more generally applicable than if we developed them for a particular risk modeling language only. In RASEN we plan to instantiate the techniques in CORAS. Another research strand in this direction is our work on extending CORAS to support component-based risk analysis and the composition of reusable risk analysis artifacts. This work also includes the development of Strict CORAS as a means to ensure that the results of different analyses are compatible and therefore easier to compose and to support by tools.

For test-based security risk assessment, we have in this deliverable explain how we use security testing to facilitate various risk assessment activities such as risk identification, risk estimation and risk treatment verification. While the RASEN process for combining security testing and security risk assessment is supported by several tools and guidelines, there is still a need for techniques for how to systematically make use of the test results when conducting the risk assessment. For this purpose we will use security indicators and risk metrics for capturing test results and aggregating these to form the input that can be used for making or updating the risk models.

Our ongoing work includes the further development of the compositional proof rules, which we will start to apply in the RASEN use cases. For the test-based risk assessment we are further developing the techniques that we need to support the process described in Section 6, and also on integrating the tools for supporting this process.

References

[1] C. J. Alberts, A. J. Dorofee. OCTAVE Criteria. Technical Report CMU/SEI-2001-TR-016, CERT (2001)

[2] I. Ben-Gal. Bayesian networks. In F. Ruggeri, R. S. Kenett, and F. W. Faltin, editors, Encyclopedia of Statistics in Quality and Reliability. John Wiley & Sons, 2007.

[3] G. Brændeland, A. Refsdal, and K. Stølen. Modular analysis and modelling of risk scenarios with dependencies. J. Syst. Softw., 83(10):1995–2013, 2010.

[4] C. Clapham, J. Nicholson: Concise Dictionary of Mathematics, third edition. Oxford University Press (2005)

[5] D. R. Comings, W. W. Ting: IA OM as an Enterprise Risk Management Metric. Chapter 10 in: Risk Management – Current Issues and Challenges, InTech (2012)

[6] European Telecommunications Standards Institute: ETSI GS ISI 001-1 v1.1.1 – Information Security Indicators (ISI); Indicators (INC); Part 1: A full set of operational indicators for organizations to use to benchmark their security posture (2013)

[7] A. Hammond, A. Adriaanse, E. Rodenburg, D. Bryant, R. Woodward: Environmental Indicators – A Systematic Approach to Measuring and Reporting on Environmental Policy Performance in the Context of Sustainable Development. World Resources Institute (1995)

[8] I. Hogganvik: A graphical approach to security risk analysis. Ph.D. thesis, University of Oslo (2007)

[9] International Electrotechnical Commission. IEC 61025 Fault Tree Analysis (FTA), 1990.

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

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

[12] International Organization for Standardization/International Electrotechnical Commission: ISO/IEC 27001 – Information technology – Security techniques – Information security management systems – Requirements (2005)

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

[14] International Organization for Standardization / International Electrotechnical Commission: ISO/IEC 31010 Risk management – Risk assessment techniques (2009)

[15] RASEN: Baseline for compositional test-based security risk assessment. RASEN project deliverable D3.1.1 (2013)

[16] International Organization for Standardization / International Electrotechnical Commission. ISO/IEC 10746-3 – Information technology – Open distributed processing – Reference model: Architecture, 2009.

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

[18] O. S. Ligaarden, M. S. Lund, A. Refsdal, F. Seehusen, K. Stølen: An architectural pattern for enterprise level monitoring tools. In: Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA'11), pp. 1-10. IEEE Computer Society (2011)

[19] O. S. Ligaarden, A. Refsdal, K. Stølen: Using indicators to monitor risk in interconnected systems: How to capture and measure the impact of service dependencies on the quality of provided services. Chapter in: IT Security Governance and Innovations: Theory and Research, pp. 256-292, IGI Global (2012)

[20] M. S. Lund, B. Solhaug, and K. Stølen. Model-Driven Risk Analysis – The CORAS Approach. Springer, 2011.

[21] F. Massacci, J. Mylopoulos, and N. Zannone. Security requirements engineering: The SI* modeling language and the Secure Tropos methodology. In Advances in Intelligent Information Systems, volume 265 of Studies in Computational Intelligence, pages 147–174. Springer, 2010.

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

[23] S. K. Pandey, K. Mustafa: A comparative study of risk assessment methodologies for information systems. Bulletin of Electrical Engineering and Informatics, 1(2), 111-122 (2012)

[24] 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)

[25] Risk Management Insight: FAIR (Factor Analysis of Information Risk) – Basic Risk Assessment Guide. [ONLINE] Available at: http://riskmanagementinsight.com/media/documents/FAIR_brag.pdf [Accessed 10 September 2013]

[26] B. Smith, L. Williams: On the Effective Use of Security Test Patterns, Proceedings of the Sixth International Conference on Software Security and Reliability (SERE), 2012 IEEE, pp. 108-117, DOI: 10.1109/SERE.2012.23

[27] W. Gleißner, T. Berger (2004): Auf nach Monte Carlo: Simulationsverfahren zur Risiko-Aggregation. RISKNEWS, 1: 30–37. DOI: 10.1002/risk.200490005

[28] S. Greenland (2001): Sensitivity Analysis, Monte Carlo Risk Analysis, and Bayesian Uncertainty Assessment. Risk Analysis, 21: 579–584. doi: 10.1111/0272-4332.214136

[29] A. Kolmogorov: Grundbegriffe der Wahrscheinlichkeitsrechnung, Springer Verlag Berlin 1933

[30] M. Stamatelatos, W. Vesley, J. Dugan, J. Fragola, J. Minarick, J. Railsback: Fault tree handbook with aerospace applications. Version 1.1. NASA Office of Safety and Mission Assurance (2002)

[31] 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)

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

[33] O. Caprotti and D. Carlisle. 1999. OpenMath and MathML: semantic markup for mathematics. Crossroads 6, 2 (November 1999), 11-14. DOI=10.1145/333104.333110

[34] National Vulnerability Database – NVD Common Vulnerability Scoring System Support v2. [ONLINE] Available at http://nvd.nist.gov/cvss.cfm?version=2 [Accessed 10 September 2013]

[35] L. M. S. Tran, B. Solhaug, K. Stølen: An approach to select cost-effective risk countermeasures. In Proc. 27th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSec'13). LNCS 7964, pp. 266-273, Springer (2013)

[36] L. M. S. Tran, B. Solhaug, K. Stølen: An Approach to Select Cost-Effective Risk Countermeasures Exemplified in CORAS. Technical Report A24343, SINTEF ICT (2013)

[37] B. Violino: IT risk assessment frameworks: real-world experience (2010). [ONLINE] Available at: http://www.csoonline.com/article/592525/it-risk-assessment-frameworks-real-world-experience [Accessed 10 September 2013]

[38] P. M Winter, E. Kozik: Risk Assessment Using LAVA (Los Alamos Vulnerability and Risk Assessment Methodology). Pennsylvania State University (1993)

[39] Z. Yazar: A qualitative risk analysis and management tool – CRAMM. SANS Institute (2002)

Appendix

Formal Foundations for Compositionality in Risk Assessment

Bjørnar Solhaug and Ketil Stølen

A Formal foundation

In the following we introduce the formal machinery.

A.1 Basics

$\mathbb{N}$ and $\mathbb{R}$ denote the sets of natural numbers and real numbers, respectively. We use $\mathbb{N}_0$ to denote the set of natural numbers including 0, while $\mathbb{R}^+$ denotes the set of nonnegative real numbers. This means that:

$$\mathbb{N}_0 \overset{\mathsf{def}}{=} \mathbb{N} \cup \{0\}, \quad \mathbb{R}^+ \overset{\mathsf{def}}{=} \{r \in \mathbb{R} \mid r \geq 0\}$$

For any set of elements, we use $\mathbb{P}(A)$ to denote the powerset of A.

A tuple is an element of a Cartesian product. We use π_j to extract the jth element of a tuple. Hence, if

$$(a, a') \in A \times A'$$

then $\pi_1.(a, a') = a$ and $\pi_2.(a, a') = a'$.

A.2 Sequences

By A^{∞}, A^{ω} and A^{*} we denote the set of all infinite sequences, the set of all finite and infinite sequences and the set of all finite sequences over some set of elements A, respectively. Hence, we have that

$$A^{\omega} = A^{\infty} \cup A^{*}$$

We define the functions

$$\#\_ \in A^{\omega} \to \mathbb{N}_0 \cup \{\infty\}, \quad \_[\_] \in A^{\omega} \times \mathbb{N} \to A$$

to yield the length and the nth element of a sequence. Hence, $\#s$ yields the number of elements in s, and $s[n]$ yields the nth element of s if $n \leq \#s$.

We also need functions for concatenation and filtering:

$$\_ \frown \_ \in A^{\omega} \times A^{\omega} \to A^{\omega}, \quad \_ⓢ\_ \in \mathbb{P}(A) \times A^{\omega} \to A^{\omega}$$

Concatenating two sequences implies gluing them together. Hence, $s_1 \frown s_2$ denotes a sequence of length $\#s_1 + \#s_2$ that equals s_1 if s_1 is infinite, and is prefixed by s_1 and suffixed by s_2, otherwise.

The filtering operator is used to filter away elements. $B ⓢ s$ denotes the subsequence obtained from s by removing all elements in s that are not in the set B.

A.3 Timed events

$\mathbb{E}$ denotes the set of all events, while the set of all timestamps is defined by

$$\mathbb{T} \stackrel{\mathsf{def}}{=} \mathbb{R}^+$$

A timed event is an element of

$$\mathbb{E} \times \mathbb{T}$$

A.4 Histories

A history is an infinite sequence of timed events that is ordered by time and progresses beyond any finite point in time. Hence, a history is an element of:[1]

$$\begin{aligned} \mathbb{H} \stackrel{\mathsf{def}}{=} \{ \quad & h \in (\mathbb{E} \times \mathbb{T})^{\infty} \mid \\ & \quad \forall\, n \in \mathbb{N} : \pi_2.h[n] \leq \pi_2.h[n+1] \\ & \quad \forall\, t \in \mathbb{T} : \exists\, n \in \mathbb{N} : \pi_2.h[n] > t \quad \} \end{aligned}$$

The first conjunct requires the timestamp of a timed event to be at least as great as that of its predecessor. The second conjunct makes sure that time will always progress beyond any finite point in time. That is, for any timestamp t and history h there is a timed event in h whose timestamp is greater than t.

We also need a function for truncating histories

$$\_|\_ \in \mathbb{H} \times \mathbb{T} \to (\mathbb{E} \times \mathbb{T})^{*}$$

The truncation operator captures the prefix of a history up to and including a certain point in time. Hence, $h|_t$ describes the maximal prefix of h whose timed events all have timestamps less than or equal to t.

A.5 Frequencies

As explained above, we use the nonnegative real numbers to represent time. The time unit is equal to 1. For simplicity, we assume that all frequencies are per time unit. The set of frequencies F is therefore defined as follows:

$$\mathbb{F} \stackrel{\mathsf{def}}{=} \mathbb{R}^+$$

Hence, $f \in \mathbb{F}$ denotes the frequency of f occurrences per time unit.

B Risk graphs

B.1 Syntax of risk graph formulas

B.1.1 Risk graphs

A risk graph is a pair of two sets (V, R) where

$$V \subseteq \mathbb{P}(\mathbb{E}) \times \mathbb{F}, \quad R \subseteq V \times \mathbb{R}^+ \times V$$

We refer to the elements of V as vertices and to the elements of R as relations. We use $v(f)$ to denote a vertex, while $v \xrightarrow{r} v'$ denotes a relation.

[1]We often use indentation to represent conjunction.

B.1.2 Vertex expressions

The set of vertex expressions is the smallest set X_V such that

$$\mathbb{P}(\mathbb{E}) \subseteq X_V, \quad v, v' \in X_V \Rightarrow v \sqcup v' \in X_V \wedge v \sqcap\!| \, v' \in X_V$$

We need a function

$$s \in X_V \rightarrow \mathbb{P}(\mathbb{E})$$

that for any vertex expression yields its set of events. Formally, s is defined recursively as follows:

$$s(v) \overset{\mathsf{def}}{=} \begin{cases} v & \text{if } v \in \mathbb{P}(\mathbb{E}) \\ s(v_1) \cup s(v_2) & \text{if } v = v_1 \sqcup v_2 \\ s(v_2) & \text{if } v = v_1 \sqcap\!| \, v_2 \end{cases}$$

B.1.3 Risk graph formula

A risk graph formula is of one of the following two forms

$$H \vdash v(f), \quad H \vdash v \xrightarrow{r} v'$$

where

- $H \in \mathbb{P}(\mathbb{H}) \setminus \varnothing$,
- $v, v' \in X_V$,
- $f \in \mathbb{F}$,
- $r \in \mathbb{R}^{+}$.

B.2 Semantics of risk graph formulas

We use the brackets $[\![\ ]\!]$ to extract the semantics of a risk graph formula. If $v \in \mathbb{P}(\mathbb{E})$ we define:

$$\begin{aligned} &[\![\ H \vdash v(f) \ ]\!] \overset{\mathsf{def}}{=} \\ &\qquad \forall\, h \in H : \\ &\qquad\qquad f = \lim_{t \rightarrow \infty} \tfrac{\#((v \times \mathbb{T}) \circledS (h|_t))}{t} \end{aligned}$$

The semantics of any other risk graph formula is defined recursively as follows:

$$\begin{aligned} &[\![\ H \vdash v_1 \sqcup v_2(f) \ ]\!] \overset{\mathsf{def}}{=} \\ &\qquad \exists\, f_1, f_2, f_3 \in \mathbb{F} : \\ &\qquad\qquad [\![\ H \vdash v_1(f_1) \ ]\!] \\ &\qquad\qquad [\![\ H \vdash v_2(f_2) \ ]\!] \\ &\qquad\qquad [\![\ H \vdash s(v_1) \cap s(v_2)(f_3) \ ]\!] \\ &\qquad\qquad f_1 + f_2 - f_3 \leq f \leq f_1 + f_2 \end{aligned}$$

$$
\begin{aligned}
&[\![\ H \vdash v_1 \sqcap\!\!| \ v_2(f)\ ]\!] \overset{\mathsf{def}}{=} \\
&\qquad \exists\, r \in \mathbb{R}^+;\ f_1, f_2 \in \mathbb{F} : \\
&\qquad\qquad [\![\ H \vdash v_1(f_1)\ ]\!] \\
&\qquad\qquad [\![\ H \vdash v_2(f_2)\ ]\!] \\
&\qquad\qquad f = f_1 \cdot r \\
&\qquad\qquad f \leq f_2
\end{aligned}
$$

$$
\begin{aligned}
&[\![\ H \vdash v_1 \xrightarrow{r} v_2\ ]\!] \overset{\mathsf{def}}{=} \\
&\qquad \exists f_1, f_2 \in \mathbb{F} : \\
&\qquad\qquad [\![\ H \vdash v_1(f_1)\ ]\!] \\
&\qquad\qquad [\![\ H \vdash v_2(f_2)\ ]\!] \\
&\qquad\qquad f_2 \geq f_1 \cdot r
\end{aligned}
$$

B.3 Calculus of risk graph formulas

The three rules below correspond to rules 13.10, 13.11 and 13.12 in the CORAS book, respectively. There are some minor differences. In the CORAS book the real number decorating a leads-to relation is restricted to $[0, 1]$. The statistical independence constraint in Rule 13.12 of the CORAS book is not needed.

B.3.1 Rule for leads-to

$$
\frac{H \vdash v_1(f) \quad H \vdash v_1 \xrightarrow{r} v_2}{H \vdash v_1 \sqcap\!\!|\ v_2(f \cdot r)}
$$

Soundness Assume

(1) $H \vdash v_1(f)$

(2) $H \vdash v_1 \xrightarrow{r} v_2$

Then

(3) $H \vdash (v_1 \sqcap\!\!|\ v_2)(f \cdot r)$

Proof: (2) implies there are $f_1, f_2 \in \mathbb{F}$ such that

(4) $[\![\ H \vdash v_1(f_1)\ ]\!]$

(5) $[\![\ H \vdash v_2(f_2)\ ]\!]$

(6) $f_2 \geq f_1 \cdot r$

(1) and (4) imply

(7) $f = f_1$

(6) and (7) imply

(8) $f_2 \geq f \cdot r$

(4), (5), (7) and (8) imply (3).

B.3.2 Rule for mutually exclusive vertices

$$\frac{H_1 \vdash v_1(f) \wedge v_2(0) \quad H_2 \vdash v_2(f) \wedge v_1(0)}{H_1 \cup H_2 \vdash v_1 \sqcup v_2(f)}$$

For simplicity we have merged four premises into two using logical conjunction.[2]

Soundness Assume

(1) $H_1 \vdash v_1(f) \wedge v_2(0)$
(2) $H_2 \vdash v_2(f) \wedge v_1(0)$

Then

(3) $H_1 \cup H_2 \vdash v_1 \sqcup v_2(f)$

Proof: (1) and (2) imply

(4) $H_1 \cap H_2 = \varnothing \vee f = 0$

(1) and (2) imply

(5) $H_1 \vdash v_1 \sqcup v_2(f)$
(6) $H_2 \vdash v_1 \sqcup v_2(f)$

(4), (5) and (6) imply (3).

B.3.3 Rule for separate vertices

$$\frac{H \vdash v_1(f_1) \quad H \vdash v_2(f_2) \quad s(v_1) \cap s(v_2) = \varnothing}{H \vdash v_1 \sqcup v_2(f_1 + f_2)}$$

Soundness Assume

(1) $H \vdash v_1(f_1)$
(2) $H \vdash v_2(f_2)$
(3) $s(v_1) \cap s(v_2) = \varnothing$

Then

(4) $H \vdash v_1 \sqcup v_2(f_1 + f_2)$

Proof: (3) implies

(5) $H \vdash s(v_1) \cap s(v_2)(0)$

(1), (2), (5) and the fact that $f_1 + f_2 - 0 \leq f_1 + f_2 \leq f_1 + f_2$ imply (4).

[2]Hence, $H \vdash X \wedge Y$ means $H \vdash X$ and $H \vdash Y$.

C Introducing countermeasures

C.1 Formal foundation extended with countermeasures

We start by extending the basic formal machinery to take countermeasures into consideration.

C.1.1 Timed events with countermeasures

$\mathbb{C}$ denotes the set of all countermeasures. To record treatments each timed event is extended with a possibly empty set of countermeasures. A timed event with an empty set of countermeasures is untreated, while a timed event with a nonempty set is treated by the countermeasures in the set. Hence, a timed event is from this point onwards an element of

$$\mathbb{E} \times \mathbb{T} \times \mathbb{P}(\mathbb{C})$$

C.1.2 Histories with countermeasures

The notion of history is generalized straightforwardly to deal with timed events with countermeasures as follows:

$$\begin{aligned} \mathbb{H} \overset{\mathsf{def}}{=} \{ \quad & h \in (\mathbb{E} \times \mathbb{T} \times \mathbb{P}(\mathbb{C}))^{\infty} \mid \\ & \quad \forall\, n \in \mathbb{N} : \pi_2.h[n] \leq \pi_2.h[n+1] \\ & \quad \forall\, t \in \mathbb{T} : \exists\, n \in \mathbb{N} : \pi_2.h[n] > t \quad \} \end{aligned}$$

The truncation operator

$$\_|\_ \in \mathbb{H} \times \mathbb{T} \rightarrow (\mathbb{E} \times \mathbb{T} \times \mathbb{P}(\mathbb{C}))^{*}$$

is generalized accordingly.

C.2 Syntax extended with countermeasures

The next step is to generalize the notion of risk graph.

C.2.1 Risk graphs

A risk graph with treatments is a tuple of five sets (V, C, R_l, R_e, R_d) where

$$\begin{aligned} & V \subseteq \mathbb{P}(\mathbb{E}) \times \mathbb{F}, \\ & C \subseteq \mathbb{C}, \\ & R_l \subseteq V \times \mathbb{R}^{+} \times V, \\ & R_e \subseteq C \times [0,1] \times V, \\ & R_d \subseteq C \times [0,1] \times R_e \end{aligned}$$

We refer to the elements of V as the set of vertices, C as the set of countermeasures, and to R_l, R_e, R_d as the leads-to relations, the effects relations and the dependency relations, respectively.

We use $v(f)$ to denote a vertex, c to denote a countermeasure, $\xrightarrow{l}$ to denote a leads-to relation, $\xrightarrow{e}$ to denote an effects relation and $\xrightarrow{d}$ to denote a dependency relation.

C.2.2 Vertex expressions

The set of vertex expressions is the smallest set X_V such that

$$v \in \mathbb{P}(\mathbb{E}) \wedge cs \in \mathbb{P}(\mathbb{C}) \Rightarrow v_{cs} \in X_V$$
$$v, v' \in X_V \Rightarrow v \sqcup v' \in X_V \wedge v \sqcap\!\!| \; v' \in X_V$$

We need a function

$$s \in X_V \rightarrow \mathbb{P}(\mathbb{E})$$

that for any vertex expression calculates its set of events. Formally, s is defined recursively as follows:

$$s(v) \overset{\mathsf{def}}{=} \begin{cases} v' & \text{if } v = v'_{cs} \\ s(v_1) \cup s(v_2) & \text{if } v = v_1 \sqcup v_2 \\ s(v_2) & \text{if } v = v_1 \sqcap\!\!| \; v_2 \end{cases}$$

C.2.3 Risk graph formula

A risk graph formula is of one of the following four forms

$$H \vdash c \xrightarrow{e}_{cs} v, \quad H \vdash c \xrightarrow{d} (c' \xrightarrow{e}_{cs} v), \quad H \vdash v'(f), \quad H \vdash v' \xrightarrow{r} v''$$

where

- $H \in \mathbb{P}(\mathbb{H})$,
- $c, c' \in \mathbb{C}$ where $c \neq c'$,
- $e, d \in [0, 1]$,
- $cs \in \mathbb{P}(\mathbb{C})$ where $c, c' \notin cs$,
- $v \in \mathbb{P}(\mathbb{E})$,
- $v', v'' \in X_V$,
- $f \in \mathbb{F}$,
- $r \in \mathbb{R}^+$.

C.3 Semantics extended with countermeasures

The semantics of a risk graph formula is defined recursively as before. In particular, the definitions are unchanged in the case of

$$[\![\; H \vdash v_1 \sqcup v_2(f) \;]\!], \quad [\![\; H \vdash v_1 \sqcap\!\!| \; v_2(f) \;]\!], \quad [\![\; H \vdash v_1 \xrightarrow{r} v_2 \;]\!]$$

The vertex base-case must however be updated to take countermeasures into account:

$$[\![\; H \vdash v_{cs}(f) \;]\!] \overset{\mathsf{def}}{=}$$
$$\forall\, h \in H :$$
$$f = \lim_{t \to \infty} \tfrac{\#((v \times \mathbb{T} \times \mathbb{P}(\mathbb{C} \setminus cs)) \, Ⓢ \, (h|_t))}{t}$$

Hence, we only take into consideration those events in v that are not treated by a countermeasure in cs.

In the case of the effects relation the semantics is defined as follows:

$$
\begin{aligned}
&[\![\ H \vdash c \xrightarrow{e}_{cs} v\ ]\!] \ \overset{\mathsf{def}}{=} \\
&\quad \exists f_1, f_2 \in \mathbb{F} : \\
&\qquad [\![\ H \vdash v_{cs}(f_1)\ ]\!] \\
&\qquad [\![\ H \vdash v_{cs\cup\{c\}}(f_2)\ ]\!] \\
&\qquad f_1 \neq 0 \Rightarrow e = \tfrac{f_1 - f_2}{f_1}
\end{aligned}
$$

Hence, e is the fraction of v events whose set of countermeasures contains c but no countermeasure in cs.

Also the dependency relation captures a fraction:

$$
\begin{aligned}
&[\![\ H \vdash c \xrightarrow{d} (c' \xrightarrow{e}_{cs} v)\ ]\!] \ \overset{\mathsf{def}}{=} \\
&\quad [\![\ H \vdash c' \xrightarrow{e}_{cs} v\ ]\!] \Rightarrow \\
&\qquad \exists\, e' \in [0,1] : \\
&\qquad\quad [\![\ H \vdash c' \xrightarrow{e'}_{cs\cup\{c\}} v\ ]\!] \\
&\qquad\quad e \neq 0 \Rightarrow d = 1 - \tfrac{e'}{e}
\end{aligned}
$$

Hence, d is the fraction of v events treated by countermeasure c' that is also treated by countermeasure c.

C.4 Calculus extended with countermeasures

C.4.1 Rule for countermeasure effect

$$
\frac{H \vdash c \xrightarrow{e}_{cs} v \quad H \vdash v_{cs}(f)}{H \vdash v_{cs\cup\{c\}}(f \cdot \overline{e})}
$$

Soundness Assume

(1) $H \vdash c \xrightarrow{e}_{cs} v$

(2) $H \vdash v_{cs}(f)$

Then

(3) $H \vdash v_{cs\cup\{c\}}(f \cdot \overline{e})$

Proof: (1) implies there are $f_1, f_2 \in \mathbb{F}$ such that

(4) $[\![\ H \vdash v_{cs}(f_1)\ ]\!]$

(5) $[\![\ H \vdash v_{cs\cup\{c\}}(f_2)\ ]\!]$

(6) $f_1 \neq 0 \Rightarrow e = \frac{f_1 - f_2}{f_1}$

(2) and (4) imply

(7) $f = f_1$

There are two cases to consider:

- Assume

  (8) $f_1 = 0$

  (4), (7) and (8) imply

  (9) $[\![\ H \vdash v_{cs \cup \{c\}}(0)\ ]\!]$

  (7) and (8) imply

  (10) $f = 0$

  (9), (10) and $0 \cdot \overline{e} = 0$ imply (3).

- Assume

  (11) $f_1 \neq 0$

  (6), (7) and (11) imply

  (12) $e = \frac{f - f_2}{f}$

  (12) implies

  (13) $\frac{f_2}{f} = 1 - e$

  (13) implies

  (14) $f_2 = f \cdot \overline{e}$

  (5) and (14) imply (3).

C.4.2 Rule for countermeasure dependency

$$\frac{H \vdash c \xrightarrow{d} (c' \xrightarrow{e}_{cs} v) \quad H \vdash c' \xrightarrow{e}_{cs} v}{H \vdash c' \xrightarrow{e \cdot \overline{d}}_{cs \cup \{c\}} v}$$

Soundness Assume

(1) $H \vdash c \xrightarrow{d} (c' \xrightarrow{e}_{cs} v)$
(2) $H \vdash c' \xrightarrow{e}_{cs} v$

Then

(3) $H \vdash c' \xrightarrow{e \cdot \overline{d}}_{cs \cup \{c\}} v$

Proof: There are two cases to consider:

- Assume

  (4) $e \neq 0$

  (1), (2) and (4) imply there is $e' \in [0,1]$ such that

  (5) $[\![\; H \vdash c' \xrightarrow{e'}_{cs \cup \{c\}} v \;]\!]$
  (6) $d = 1 - \frac{e'}{e}$

  (6) implies

  (7) $\frac{e'}{e} = 1 - d = \overline{d}$

  (5) and (7) imply (3).

- Assume

  (8) $e = 0$

  (2) implies there are $f_1, f_2 \in \mathbb{F}$ such that

  (9) $[\![\; H \vdash v_{cs}(f_1) \;]\!]$
  (10) $[\![\; H \vdash v_{cs \cup \{c'\}}(f_2) \;]\!]$
  (11) $f_1 \neq 0 \Rightarrow e = \frac{f_1 - f_2}{f_1}$

  Again, there are two cases to consider:

  – Assume

    (12) $f_1 = 0$

    (9) and (12) imply

    (13) $[\![\; H \vdash v_{cs \cup cs'}(0) \;]\!]$

    for arbitrary cs'. This implies (3).

  – Assume

    (14) $f_1 \neq 0$

    (8), (11) and (14) imply that $f_1 = f_2$ which means that the treatment c' has no effect in addition to the effect of cs. This implies (3).

D Introducing consequences

D.1 Formal foundation extended with consequences

We start by extending the basic formal machinery to take consequences into consideration.

D.1.1 Timed events with consequences

$\mathbb{I}$ denotes the set of all consequences (or impacts). To facilitate arithmetic operations on consequences we assume that

$$\mathbb{I} \overset{\mathsf{def}}{=} \mathbb{R}^+$$

To record consequences each timed event is extended with an additional component characterizing the consequence of this event with respect to the various combinations of countermeasures. A timed event is from this point onwards an element of

$$\mathbb{E} \times \mathbb{T} \times \mathbb{P}(\mathbb{C}) \times (\mathbb{P}(\mathbb{C}) \to \mathbb{I})$$

For any timed event e we require

$$c \subseteq c' \Rightarrow (\pi_4.e)(c) \geq (\pi_4.e)(c')$$

Hence, adding a countermeasure will never increase the consequence.

D.1.2 Histories with consequences

The notion of history is generalized straightforwardly to deal with consequences as follows:

$$\begin{aligned}\mathbb{H} \overset{\mathsf{def}}{=} \{ \quad & h \in (\mathbb{E} \times \mathbb{T} \times \mathbb{P}(\mathbb{C}) \times (\mathbb{P}(\mathbb{C}) \to \mathbb{I}))^{\infty} \mid \\ & \forall\, n \in \mathbb{N} : \pi_2.h[n] \leq \pi_2.h[n+1] \\ & \forall\, t \in \mathbb{T} : \exists\, n \in \mathbb{N} : \pi_2.h[n] > t \qquad \}\end{aligned}$$

The truncation operator

$$\_|\_ \in \mathbb{H} \times \mathbb{T} \to (\mathbb{E} \times \mathbb{T} \times \mathbb{P}(\mathbb{C}) \times (\mathbb{P}(\mathbb{C}) \to \mathbb{I}))^{*}$$

is generalized accordingly.

D.2 Syntax extended with consequences

The next step is to generalize the notion of risk graph.

D.2.1 Risk graphs

The notion of risk graph is a tuple of five sets (V, C, R_l, R_e, R_d) where

$$\begin{aligned}& V \subseteq \mathbb{P}(\mathbb{E}) \times \mathbb{F} \times \mathbb{I}, \\ & C \subseteq \mathbb{C}, \\ & R_l \subseteq V \times \mathbb{R}^+ \times V, \\ & R_e \subseteq C \times [0,1] \times [0,1] \times V, \\ & R_d \subseteq C \times [0,1] \times [0,1] \times R_e\end{aligned}$$

We use $v(f, i)$ to denote a vertex, $\xrightarrow{(e_f, e_i)}$ to denote an effects relation and $\xrightarrow{(d_f, d_i)}$ to denote a dependency relation. The remaining conventions are as before.

D.2.2 Vertex expressions

The notion of vertex expression is left unchanged.

D.2.3 Risk graph formula

A risk graph formula is of one of the following four forms

$$H \vdash c \xrightarrow{(e_f,e_i)}_{cs} v, \quad H \vdash c \xrightarrow{(d_f,d_i)} (c' \xrightarrow{(e_f,e_i)}_{cs} v), \quad H \vdash v'(f,i), \quad H \vdash v' \xrightarrow{r} v''$$

where

- $H \in \mathbb{P}(\mathbb{H})$,
- $c, c' \in \mathbb{C}$ where $c \neq c'$,
- $e_f, e_i, d_f, d_i \in [0,1]$,
- $cs \in \mathbb{P}(\mathbb{C})$ where $c, c' \notin cs$,
- $v \in \mathbb{P}(\mathbb{E})$,
- $v', v'' \in X_V$,
- $f \in \mathbb{F}$,
- $i \in \mathbb{I}$,
- $r \in \mathbb{R}^+$.

D.3 Semantics extended with consequences

```
$[\![ H \vdash v_{cs}(f,i) ]\!] \overset{\mathsf{def}}{=}$
    $\forall\, h \in H :$
        let
            $x = (v \times \mathbb{T} \times \mathbb{P}(\mathbb{C} \setminus cs) \times (\mathbb{P}(\mathbb{C}) \to \mathbb{I})) \circledS h$
        in
            $\#x = 0 \Rightarrow$
                $f = 0$
                $i = 0$
            $\#x \neq 0 \Rightarrow$
                $f = \lim_{t \to \infty} \frac{\#(x|_t)}{t}$
                $i = \lim_{t \to \infty} \frac{\sum_{1 \leq j \leq \#(x|_t)} \pi_4.x[j](cs)}{\#(x|_t)}$

$[\![ H \vdash v_1 \sqcup v_2(f,i) ]\!] \overset{\mathsf{def}}{=}$
    $\exists f_1, f_2, f_3 \in \mathbb{F} :$
        $[\![ H \vdash v_1(f_1,i) ]\!]$
        $[\![ H \vdash v_2(f_2,i) ]\!]$
        $[\![ H \vdash s(v_1) \cap s(v_2)(f_3,i) ]\!]$
        $f_1 + f_2 - f_3 \leq f \leq f_1 + f_2$
```

$$
\begin{array}{l}
[\![\ H \vdash v_1 \sqcap| \, v_2(f, i)\ ]\!] \overset{\mathsf{def}}{=} \\
\quad \exists\, r \in \mathbb{R}^+;\ f_1, f_2 \in \mathbb{F};\ i' \in \mathbb{I} : \\
\qquad [\![\ H \vdash v_1(f_1, i')\ ]\!] \\
\qquad [\![\ H \vdash v_2(f_2, i)\ ]\!] \\
\qquad f = f_1 \cdot r \\
\qquad f \leq f_2
\end{array}
$$

$$
\begin{array}{l}
[\![\ H \vdash v_1 \xrightarrow{r} v_2\ ]\!] \overset{\mathsf{def}}{=} \\
\quad \exists f_1, f_2 \in \mathbb{F};\ i_1, i_2 \in \mathbb{I} : \\
\qquad [\![\ H \vdash v_1(f_1, i_1)\ ]\!] \\
\qquad [\![\ H \vdash v_2(f_2, i_2)\ ]\!] \\
\qquad f_2 \geq f_1 \cdot r
\end{array}
$$

$$
\begin{array}{l}
[\![\ H \vdash c \xrightarrow{(e_f, e_i)}_{cs} v\ ]\!] \overset{\mathsf{def}}{=} \\
\quad \exists f_1, f_2 \in \mathbb{F};\ i_1, i_2 \in \mathbb{I} : \\
\qquad [\![\ H \vdash v_{cs}(f_1, i_1)\ ]\!] \\
\qquad [\![\ H \vdash v_{cs \cup \{c\}}(f_2, i_2)\ ]\!] \\
\qquad f_1 \neq 0 \Rightarrow e_f = \frac{f_1 - f_2}{f_1} \\
\qquad i_1 \neq 0 \Rightarrow e_i = \frac{i_1 - i_2}{i_1}
\end{array}
$$

$$
\begin{array}{l}
[\![\ H \vdash c \xrightarrow{(d_f, d_i)} (c' \xrightarrow{(e_f, e_i)}_{cs} v)\ ]\!] \overset{\mathsf{def}}{=} \\
\quad [\![\ H \vdash c' \xrightarrow{(e_f, e_i)}_{cs} v\ ]\!] \Rightarrow \\
\qquad \exists\, e'_f, e'_i \in [0, 1] : \\
\qquad\quad [\![\ H \vdash c' \xrightarrow{(e'_f, e'_i)}_{cs \cup \{c\}} v\ ]\!] \\
\qquad\quad e_f \neq 0 \Rightarrow d_f = 1 - \frac{e'_f}{e_f} \\
\qquad\quad e_i \neq 0 \Rightarrow d_i = 1 - \frac{e'_i}{e_i}
\end{array}
$$

D.4 Calculus extended with consequences

D.4.1 Rule for leads-to

$$
\frac{H \vdash v_1(f_1, i_1) \quad H \vdash v_1 \xrightarrow{r} v_2 \quad H \vdash v_2(f_2, i_2)}{H \vdash v_1 \sqcap| \, v_2(f_1 \cdot r, i_2)}
$$

Soundness We need an additional premise to conclude that i_2 is the impact of v_2. Except for that the introduction of consequences is irrelevant for the validity of the rule. Hence, the soundness follows from the soundness of Rule B.3.1.

D.4.2 Rule for mutually exclusive vertices

$$\frac{H_1 \vdash v_1(f,i) \wedge v_2(0,i) \quad H_2 \vdash v_2(f,i) \wedge v_1(0,i)}{H_1 \cup H_2 \vdash v_1 \sqcup v_2(f,i)}$$

Soundness The introduction of consequences is irrelevant for the validity of the rule. Hence, the soundness follows from the soundness of Rule B.3.2.

D.4.3 Rule for separate vertices

$$\frac{H \vdash v_1(f_1,i) \quad H \vdash v_2(f_2,i) \quad s(v_1) \cap s(v_2) = \varnothing}{H \vdash v_1 \sqcup v_2(f_1 + f_2, i)}$$

Soundness The introduction of consequences is irrelevant for the validity of the rule. Hence, the soundness follows from the soundness of Rule B.3.3.

D.4.4 Rule for countermeasure effect

$$\frac{H \vdash c \xrightarrow{(e_f,e_i)}_{cs} v \quad H \vdash v_{cs}(f,i)}{H \vdash v_{cs\cup\{c\}}(f \cdot \overline{e_f}, i \cdot \overline{e_i})}$$

Soundness Assume

(1) $H \vdash c \xrightarrow{(e_f,e_i)}_{cs} v$
(2) $H \vdash v_{cs}(f,i)$

Then

(3) $H \vdash v_{cs\cup\{c\}}(f \cdot \overline{e_f}, i \cdot \overline{e_i})$

Proof: The soundness of the frequency deduction follows from the soundness of Rule C.4.1. Hence, we focus only on the consequence deduction. (1) implies there are $f_1, f_2 \in \mathbb{F}$ and $i_1, i_2 \in \mathbb{I}$ such that

(4) $[\![\ H \vdash v_{cs}(f_1, i_1)\ ]\!]$
(5) $[\![\ H \vdash v_{cs\cup\{c\}}(f_2, i_2)\ ]\!]$
(6) $i_1 \neq 0 \Rightarrow e_i = \frac{i_1 - i_2}{i_1}$

(2) and (4) imply

(7) $i = i_1$

There are two cases to consider:

- Assume

  (8) $i_1 = 0$

(4), (5) and (7) imply

(9) $[\![\ H \vdash v_{cs \cup \{c\}}(f_2, 0)\ ]\!]$

(7) and (8) imply

(10) $i = 0$

(9), (10) and $0 \cdot \overline{e_i} = 0$ imply (3).

- Assume

  (11) $i_1 \neq 0$

  (6), (7) and (11) imply

  (12) $e_i = \frac{i - i_2}{i}$

  (12) implies

  (13) $\frac{i_2}{i} = 1 - e_i$

  (13) implies

  (14) $i_2 = i \cdot \overline{e_i}$

  (5) and (14) imply (3).

D.4.5 Rule for countermeasure dependency

$$\frac{H \vdash c \xrightarrow{(d_f, d_i)} (c' \xrightarrow{(e_f, e_i)}_{cs} v) \quad H \vdash c' \xrightarrow{(e_f, e_i)}_{cs} v}{H \vdash c' \xrightarrow{(e_f \cdot \overline{d_f},\, e_i \cdot \overline{d_i})}_{cs \cup \{c\}} v}$$

Soundness Assume

(1) $H \vdash c \xrightarrow{(d_f, d_i)} (c' \xrightarrow{(e_f, e_i)}_{cs} v)$
(2) $H \vdash c' \xrightarrow{(e_f, e_i)}_{cs} v$

Then

(3) $H \vdash c' \xrightarrow{(e_f \cdot \overline{d_f},\, e_i \cdot \overline{d_i})}_{cs \cup \{c\}} v$

Proof: The soundness of the frequency deduction follows from the soundness of Rule D.4.5. Hence, we focus only on the consequence deduction. There are two cases to consider:

- Assume

  (4) $e_i \neq 0$

(1), (2) and (4) imply there are $e'_f, e'_i \in [0,1]$ such that

$$(5) \quad [\![\ H \vdash c' \xrightarrow{(e'_f, e'_i)}_{cs \cup \{c\}} v\ ]\!]$$
$$(6) \quad d_i = 1 - \tfrac{e'_i}{e_i}$$

(6) implies

$$(7) \quad \tfrac{e'_i}{e_i} = 1 - d_i = \overline{d_i}$$

(5) and (7) imply (3).

- Assume

  $$(8) \quad e_i = 0$$

  (2), (8) and the constraint that adding a countermeasure will never increase the consequence imply (3).

E Introducing intervals

E.1 Syntax extended with intervals

The syntax is as before with the exception that we now have intervals where we earlier had singular values.

E.2 Semantics extended with intervals

The semantics is generalized to intervals in a point-wise manner:

$$\begin{aligned}
&[\![\ H \vdash v_{cs}(F, I)\ ]\!] \overset{\mathsf{def}}{=} \\
&\qquad \forall\, h \in H;\ \exists\, f \in F;\ i \in I: \\
&\qquad\qquad [\![\ \{h\} \vdash v_{cs}(f, i)\ ]\!]
\end{aligned}$$

$$\begin{aligned}
&[\![\ H \vdash v_1 \sqcup v_2(F, I)\ ]\!] \overset{\mathsf{def}}{=} \\
&\qquad \forall\, h \in H;\ \exists\, f \in F;\ i \in I: \\
&\qquad\qquad [\![\ \{h\} \vdash v_1 \sqcup v_2(f, i)\ ]\!]
\end{aligned}$$

$$\begin{aligned}
&[\![\ H \vdash v_1 \sqcap\!\!\mid v_2(F, I)\ ]\!] \overset{\mathsf{def}}{=} \\
&\qquad \forall\, h \in H;\ \exists\, f \in F;\ i \in I: \\
&\qquad\qquad [\![\ \{h\} \vdash v_1 \sqcap\!\!\mid v_2(f, i)\ ]\!]
\end{aligned}$$

$$\begin{aligned}
&[\![\ H \vdash v_1 \xrightarrow{R} v_2\ ]\!] \overset{\mathsf{def}}{=} \\
&\qquad \forall\, h \in H;\ \exists\, r \in R: \\
&\qquad\qquad [\![\ \{h\} \vdash v_1 \xrightarrow{r} v_2\ ]\!]
\end{aligned}$$

$$\llbracket\ H \vdash c \xrightarrow{(E_F,E_I)}_{cs} v\ \rrbracket \overset{\mathsf{def}}{=}$$
$$\forall\, h \in H;\ \exists\, e_f \in E_F, e_i \in E_I :$$
$$\llbracket\ \{h\} \vdash c \xrightarrow{(e_f,e_i)}_{cs} v\ \rrbracket$$

$$\llbracket\ H \vdash c \xrightarrow{(D_F,D_I)} (c' \xrightarrow{(E_F,E_I)}_{cs} v)\ \rrbracket \overset{\mathsf{def}}{=}$$
$$\forall\, h \in H;\ \exists\, d_f \in D_F;\ d_i \in D_I;\ e_f \in E_F;\ e_i \in E_I :$$
$$\llbracket\ \{h\} \vdash c \xrightarrow{(d_f,d_i)} (c' \xrightarrow{(e_f,e_i)}_{cs} v)\ \rrbracket$$

E.3 Calculus extended with intervals

E.3.1 Rule for leads-to

$$\frac{H \vdash v_1(F_1, I_1) \quad H \vdash v_1 \xrightarrow{R} v_2 \quad H \vdash v_2(F_2, I_2)}{H \vdash v_1 \sqcap\!\!| \, v_2(F_1 \cdot R, I_2)}$$

Soundness By pointwise application of Rule D.4.1.

E.3.2 Rule for mutually exclusive vertices

$$\frac{H_1 \vdash v_1(F, I) \wedge v_2(\{0\}, I) \quad H_2 \vdash v_2(F, I) \wedge v_1(\{0\}, I)}{H_1 \cup H_2 \vdash v_1 \sqcup v_2(F, I)}$$

Soundness By pointwise application of Rule D.4.2.

E.3.3 Rule for separate vertices

$$\frac{H \vdash v_1(F_1, I) \quad H \vdash v_2(F_2, I) \quad s(v_1) \cap s(v_2) = \varnothing}{H \vdash v_1 \sqcup v_2(F_1 + F_2, I)}$$

Soundness By pointwise application of Rule D.4.3.

E.3.4 Rule for countermeasure effect

$$\frac{H \vdash c \xrightarrow{(E_F,E_I)}_{cs} v \quad H \vdash v_{cs}(F, I)}{H \vdash v_{cs\cup\{c\}}(F \cdot \overline{E_F}, I \cdot \overline{E_I})}$$

Soundness By pointwise application of Rule D.4.4.

E.3.5 Rule for countermeasure dependency

$$\frac{H \vdash c \xrightarrow{(D_F,D_I)} (c' \xrightarrow{(E_F,E_I)}_{cs} v) \quad H \vdash c' \xrightarrow{(E_F,E_I)}_{cs} v}{H \vdash c' \xrightarrow{(E_F \cdot \overline{D_F}, E_I \cdot \overline{D_I})}_{cs\cup\{c\}} v}$$

Soundness By pointwise application of Rule D.4.5.

E.3.6 Rule for arbitrary vertices

$$\frac{H \vdash v_1(F_1, I) \quad H \vdash v_2(F_2, I)}{H \vdash v_1 \sqcup v_2([\mathsf{max}(\{\mathsf{min}(F_1), \mathsf{min}(F_2)\}), \mathsf{max}(F_1) + \mathsf{max}(F_2)], I)}$$

Soundness The upper bound corresponds to the case where the set of events of the two vertices in a history are disjoint, while the lower bound corresponds to the case where the set of events in a history belonging to one of the vertices is fully contained in the history's set of events belonging to the other vertex.

F Target composition

F.1 Composition operators

In the following we formally define our composition operators $\oplus$ for the composition $\oplus(H_1, \ldots, H_n)$, $H_i \in \mathbb{P}(\mathbb{H}), i \in \{1, \ldots, n\}$. The operators we define are binary, but can be used for multiple compositions by associative and commutative properties.

F.1.1 Parallel composition

Before defining parallel composition we need to introduce a new function. For any set of pairs of elements P and pair of sequences t, by $P \circledT t$ we denote the pair of sequences obtained from t by truncating the longest sequence in t at the length of the shortest sequence in t if the two are of unequal length; for each $j \in \{1, \ldots, k\}$, where k is the length of the shortest sequence in t, selecting or deleting the two elements at index j, depending on whether the pair of these elements is in the set P. For example, we have that

$$\{(1, f), (1, g)\} \circledT (\langle 1, 1, 2, 1, 2\rangle, \langle f, f, f, g, g\rangle) = (\langle 1, 1, 1\rangle, \langle f, f, g\rangle).$$

See [1] for the formal definition.

Parallel composition $\_ \parallel \_ \in \mathbb{P}(\mathbb{H}) \times \mathbb{P}(\mathbb{H}) \to \mathbb{P}(\mathbb{H})$ is now defined as follows.

$$\begin{aligned} H_1 \parallel H_2 \overset{\mathsf{def}}{=} \{h \in \mathbb{H} \mid \exists\, s \in \{1, 2\}^{\infty} : \pi_2(\{1\} \times (\mathbb{E} \times \mathbb{T})) \circledT (s, h) \in H_1 \wedge \\ \pi_2(\{2\} \times (\mathbb{E} \times \mathbb{T})) \circledT (s, h) \in H_2\} \end{aligned}$$

Parallel composition is associative and commutative.

F.1.2 Sequential composition

Sequential composition of two sets of histories is the pairwise concatenation of histories. Recall that the set of histories $\mathbb{H}$ are infinite sequences. Executions that terminate are therefore represented by histories h whose events at one point and beyond are terminating events τ. When the histories h_1 and h_2 are concatenated and the former terminates, the former is truncated at the last event before the terminating event prior to concatenation. The timestamps of the latter are incremented with the timestamp of the last event before the first terminating event of the former.

The function $\mathsf{term}(\_) \in \mathbb{H} \rightarrow \mathbb{T}$ yields the timestamp of the terminating event of a history; the function is undefined when there is no terminating event. The function $\mathsf{inc}(\_,\_) \in \mathbb{H} \times \mathbb{R}^{+} \rightarrow \mathbb{H}$ increases all timestamps in a trace with a positive real number.

Sequential composition $\_ ; \_ \in \mathbb{P}(\mathbb{H}) \times \mathbb{P}(\mathbb{H}) \rightarrow \mathbb{P}(\mathbb{H})$ is now defined as follows.

$$\begin{aligned} H_1;\ H_2 \overset{\mathsf{def}}{=} \{h \in \mathbb{H} \mid\ & \exists\, h_1 \in H_1, h_2 \in H_2 : \\ & h = h_1 \ \textit{when}\ \mathsf{term}(h_1)\ \textit{is undefined} \\ & h = h_1|_{\mathsf{term}(h_1)} \frown \mathsf{inc}(h_2, \mathsf{term}(h_1))\ \textit{otherwise}\} \end{aligned}$$

Sequential composition is associative.

F.1.3 Non-deterministic choice

Non-deterministic choice $\_ \mathbin{[\![} \_ \in \mathbb{P}(\mathbb{H}) \times \mathbb{P}(\mathbb{H}) \rightarrow \mathbb{P}(\mathbb{H})$ is defined as follows

$$H_1 \mathbin{[\![} H_2 \overset{\mathsf{def}}{=} H_1 \cup H_2$$

This composition operator is associative and commutative.

F.1.4 Features

We represent a feature F by a set of timed events, i.e. $F \in \mathbb{P}(\mathbb{E} \times \mathbb{T})$, or by a set of sub-traces, i.e. $F \in \mathbb{P}(\mathbb{H}) \setminus \varnothing$. In order to formalize the notion of features we are only missing the definition of the sub-trace relation.

For a set of elements A the sub-trace relation $\_ \lhd \_ \in A^{\omega} \times A^{\omega} \rightarrow$ *Bool* is defined as follows.

$$s_1 \lhd s_2 \overset{\mathsf{def}}{=} \exists\, s \in \{1,2\}^{\infty} : \pi_2((\{1\} \times A) \circledT (s, s_2)) = s_1$$

Whether a feature F is represented by a set of events or a set of sub-traces, we use $F \ltimes H$ to denote the target H with respect to the feature F. In both cases, the feature captures a subset of H. When F is a set of events, $F \ltimes H$ refers to the subset of H that consists of all traces in which events from F occur. When F is a set of sub-traces, $F \ltimes H$ refers to the histories in H for which there exists a sub-trace in F.

F.2 Risk graph semantics for target composition

For parallel composition, sequential composition and non-deterministic choice, the result of composing two sets of histories is a set of histories. Hence, the risk graph semantics for these operators are as defined in the previous sections.

For a feature F, $F \ltimes H$ refers to a subset of H. We could therefore choose to define the semantics directly for this subset. However, in order to keep the information about the different features and the target systems they refer to, we need to represent them as a pair of events/sub-traces and histories in our formalism.

When F is a set of timed events, i.e. $F \in \mathbb{P}(\mathbb{E} \times \mathbb{T})$, the risk graph is a set of statements about the traces of H in which the events in F occur. We define the semantics as follows.

$$[\![\ H \vdash v(f) \ ]\!] \overset{\mathsf{def}}{=}$$
$$\forall\, h \in H : (\exists\, n \in \mathbb{N}, e \in F : h[n] = e \Rightarrow f = \lim_{t \to \infty} \tfrac{\#((v \times \mathbb{T}) \circledS (h|_t))}{t})$$

When F is a set of traces, the risk graph is a set of statements about the traces of H that have sub-traces in F. The definition is as follows.

$$[\![\ H \vdash v(f) \ ]\!] \overset{\mathsf{def}}{=} \forall\, h \in H : (\exists\, h' \in F : h' \lhd h \Rightarrow f = \lim_{t \to \infty} \tfrac{\#((v \times \mathbb{T}) \circledS (h|_t))}{t})$$

References

[1] F. Seehusen, B. Solhaug, and K. Stølen. Adherence preserving refinement of trace-set properties in STAIRS: Exemplified for information flow properties and policies. *Software and Systems Modeling*, 8(1):45–65, 2009.