Towards Integration of Compositional Risk Analysis Using Monte Carlo Simulation and Security Testing

Johannes Viehmann

Fraunhofer FOKUS
Kaiserin-Augusta-Allee 31
D-10589 Berlin
Germany
Johannes.Viehmann@Fokus.Fraunhofer.de

Abstract. This short paper describes ongoing efforts to combine concepts of security risk analysis with security testing into a single process. Using risk analysis artefact composition and Monte Carlo simulation to calculate likelihood values, the method described here is intended to become applicable for complex large scale systems with dynamically changing probability values.

Keywords: Risk assessment, security testing, Monte Carlo simulation

1 Introduction

Security is crucial in various market sectors, including IT, health, aviation and aerospace. In the real world perfect security often cannot be achieved. Trust allows human beings to take remaining risks. Before trusting, before taking risks, it is reasonable to carefully analyze the chances, the potential benefits and the potential losses as far as possible. For technical systems, services and applications such an analysis might include risk assessment and security testing.

Those offering security critical technical systems, applications or services can benefit from careful risk analysis and security testing in two ways: They can use the results to treat potential weaknesses in their products. Additionally, they can use the results to communicate the identified remaining risks honestly, which can be very important to create trust.

This paper introduces new concepts to integrate compositional risk assessment and security testing into a single process. Furthermore, ideas for increasing the reusability of the risk analysis and security testing artefacts are presented.

Implementing the described methodology in a tool in order to make it practically applicable for large scale systems for which manual analysis is not practicable is currently ongoing work. The methodology and its implementation are going to be evaluated by using them in two different case studies to analyze critical systems and by comparing the results in relation to the effort with other concepts and tools for risk assessment and security testing.

2 The problems

There is little doubt that security critical technical systems should be carefully analyzed. However, both, risk assessment and security testing might be difficult and expensive.

Typically risk assessment is performed at a high level of abstraction and results depend on the experience and on subjective judgment of the analysts. Hence, results might be imprecise, unreliable and uncertain.

In contrast to risk assessment, security testing does produce objective and precise results – but only for those things that are actually tested. Even for small systems, complete testing is usually not possible since it would take too long and it would be by far too expensive. Selecting test cases that should be tested while ignoring other potential test cases is a critical step. Even highly insecure system can produce lots of correct test verdicts if the "wrong" test cases have been created and executed.

The larger and the more complex a system is, the more different components from different suppliers it contains, the harder it gets to perform a risk analysis and to do efficient security testing without losing something in between. Often there will not be experts available that were capable to do a risk assessment or security test case selection for the entire system, but only for some components they are familiar with.

3 State of the art

3.1 Compositional Risk Assessment

Risk assessment means to identify, analyze and evaluate risks which threaten assets [1] [2]. There are lots of different methods and technologies established for risk assessment, including fault tree analysis (FTA) [4], event tree analysis ETA [5], Failure Mode Effect (and Criticality) Analysis FMEA/FMECA [3] and the CORAS method [6].

However, most traditional risk assessment technologies analyze systems as a whole [7]. They do not offer support for compositional risk assessment. Compositional risk assessment combines risk analysis results for components of a complex modular system to derive a risk picture for the entire complex system without looking further into the details of its components. Nevertheless, for the mentioned risk assessment concepts, there are some publications dealing with compositional risk analysis, e.g. [8] for FTA and [9] for FMEA. In this paper, the extension for CORAS suggested in [10] will be used and developed further as the method for compositional risk assessment.

Monte Carlo simulations are widely used to analyze complex systems and especially for risk aggregation [15] [16]. They are used in the approach described here to calculate likelihood values for complex systems with dynamic dependencies.

3.2 Security Testing in Combination with Risk Assessment

There are basically two different ways how model-based security testing and security risk analysis can be combined [11]. Test-driven Security Risk Assessment tries to improve the security risk analysis with the help of security risk testing and the final output results are risk analysis artefacts. There have been several publications about this approach, e.g. [12] [13], but there is no general applicable methodology and not much tool support.

In contrast to Test-driven Security Risk Assessment, the Risk-driven Security Testing approach tries to improve the security testing with the help of security risk analysis and the final results are test result reports. There are lots of different methods, some trying to identify test cases while others try to prioritize test cases or to do both. For example, [14] uses fault trees as the starting point for identifying test cases.

[12] uses a combination of both approaches, Test-driven Security Risk Assessment and Risk-driven Security Testing, but it does not propose any technique or detailed guideline for how to update the risk model based on the test results. In this paper, another combined approach will be presented together with a methodology specifying how it should be done.

4 Compositional risk analysis with Monte Carlo simulation and security testing for selected components

4.1 Increasing the Reusability of Risk Analysis Artefacts using *Strict CORAS*

From the various existing methods and concepts for risk assessment, we choose the model based CORAS method as the starting point for our development because it is very flexible and it offers an intuitive graphic representation for its risk models. Since the CORAS method itself does not provide much support for component based and compositional risk analysis, we also use the extension for the CORAS method suggested in [10], which is designed exactly to deal with the component based compositional risk analysis of complex systems. It basically uses *Threat Interfaces* and gates in *Threat Composition Diagrams* to model dependencies, i.e. how vulnerabilities could be affected by unwanted incidents of other components. Working only with constant probability values according to the Kolmogorov axioms [17] for some fixed period of time to express likelihoods, it is relatively easy to calculate precise dependent likelihood values. But this approach is not appropriate to model complex dynamic systems in which likelihoods might change over time, for example.

CORAS itself allows the analysts to define their custom formats and scales to express likelihood values. Hence, it is more flexible than the extension for compositional risk assessment in that aspect. There is a good reason for the restriction in the extension for compositional risk assessment: Using more complex notations of likelihood values, it would become much harder or even impossible to calculate precise likelihood values for dependent incidents. Additionally, this flexibility might become a problem if risk analysis artefacts using different scales and formats should be composed as parts of the same system. Tools that should support the analyst e.g. by calculating likelihood values would have to deal with an infinite number of different constructs. The simplest solution to avoid compatibility issues and to increase reusability would be to provide default standard scales and formats that have to be used by anyone. However, it will be difficult to find a single format for expressing likelihoods that is perfectly suitable for each possible scenario. If it is powerful and flexible enough to express all important aspects, it will probably be unintuitive and too complicated for the analysis of simple systems. Hence, it is eventually a much better idea to use a powerful standard format to express likelihoods as an internal base and for file exchange, but to allow the analysts to use more intuitive simpler formats if they need less features or less precision.

Before trying to develop a tool for compositional risk assessment, we have decided to develop such common base formats and scales. We refer to those as *Strict CORAS*. The tool will then provide convertors from the simpler, more intuitive formats analysts will typically deal with to the *Strict CORAS* formats.

So what is required to express the likelihood of some single incident accurately? For real world incidents, time and dynamic changes are important factors that need to be taken into careful consideration when analyzing and describing likelihoods. For example, a probability per time period might change over time because systems might get less robust over time and failures could eventually become more likely.

Therefore, it is 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 $\mathrm{P}(T_1, T_2, T_S, L_X)$ taking a start point of time T_1, an end point of time T_2 together with two parameters T_S and L_X as arguments to express the likelihood for the occurrence of some incident in the time span 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 $\mathrm{P}(T_1, T_2, T_S, L_X)$ is going to be a XML format like OpenMath / MathML [18].

Defining a probability function $\mathrm{P}(T_1, T_2, T_S, L_X)$ might be difficult. As an example, a conversion from a simple constant probability value format like the format used in [10] to the *Strict CORAS* format is shown here in detail for independent incidents. In such a simple custom format, the likelihood for incident I is just a pair of a probability value Φ and a time period Δ for which the probability value Φ that I occurs holds. Then in *Strict CORAS* the likelihood that the incident I occurs between T_1 and T_2 can be expressed as:

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

Though calculating the probability that the incident occurs at least once, we use the likelihood that the incident does not occur (i.e. $1 - \Phi$) in the formula and finally take the inverse since there is also a certain likelihood that the incident occurs multiple times.

Sometimes, it might be easier to use frequencies instead of probability values per time period to express likelihoods. However, a frequency of 1 per time period Δ might be ambiguous. It could mean exactly one occurrence of incident I in each time period Δ. It could also mean that only in the 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 occurrence of incident I while other time periods of length Δ have multiple.

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 this can be expressed in *Strict CORAS* as shown in (2).

$$P(T_1, T_2, T_S, L_X) = \begin{cases} 1 & \text{if } \exists \text{ n} \in \mathbb{N} \mid T_2 - T_1 > T_2 - (T_S + \Omega + \Delta * n) \geq 0 \\ 0 & else \end{cases} \tag{2}$$

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 (1) with a chosen value Φ close to but slightly below one would be a good approximation using the *Strict CORAS* likelihood format.

The tool we develop is going to provide a number of formats including constant probability values and frequencies to specify likelihood values that it converts automatically to the internal *Strict CORAS* format, i.e. to probability functions.

4.2 Measuring Likelihood using Simulation and Testing

If likelihoods are expressed as *Strict CORAS* probability functions, then calculating good approximations for likelihood values of triggered incidents in compositional risk analysis using Monte Carlo simulations becomes straight forward and applicable even for complex dynamic systems. Let T_E be the point of time for which likelihood values should be evaluated. Assuming that the risk analysts have already modelled 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 and the dependencies expressed with relations and gates. For all the simulations, a common evaluation time span 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 occurs 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 S_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 modelled as a treatment occurs. **Fig. 1** shows an example.

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) after *N* simulation steps is calculated. This mean value is then a good approximation for the aggregated probability $P(T_E)$ that incident *I* occurs at time T_E.

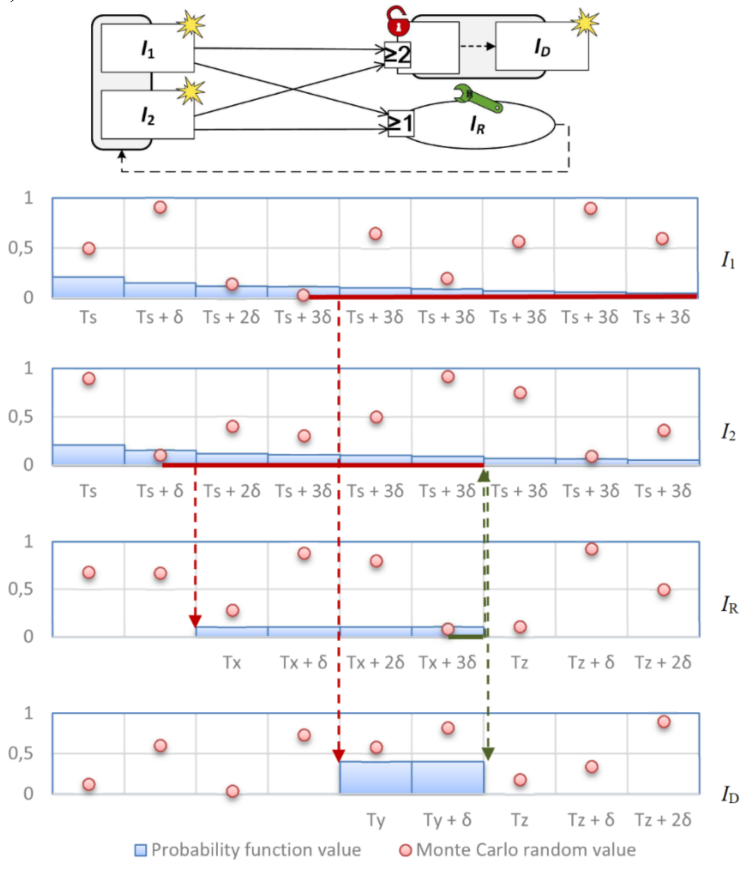


Fig. 1. *Threat Composition Diagram* with visualization of a Monte Carlo simulation for calculating whether incident I_D occurs or not. T_Y and T_Z are passed as an element of L_X to the probability function of incident I_D just like T_X and T_Z are passed to the probability function of I_R.

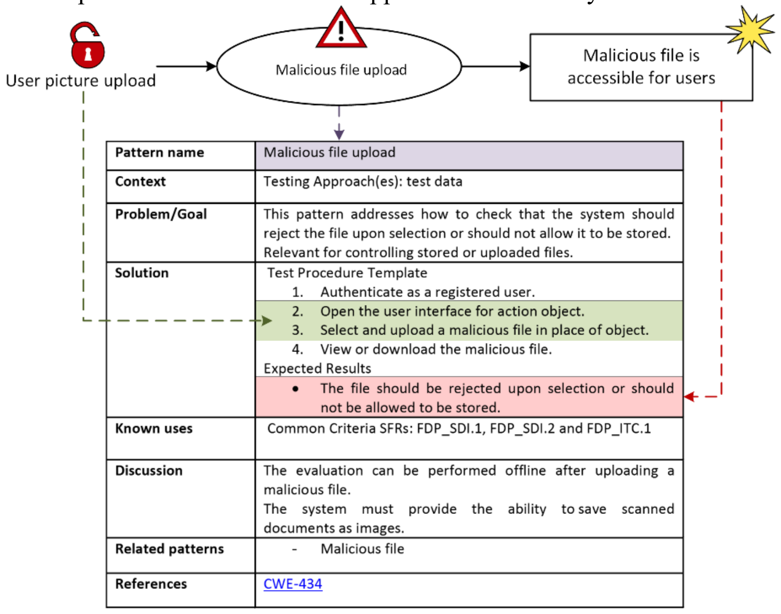
With Monte Carlo simulation, it becomes possible to calculate likelihood values even for complex systems with dynamically changing probabilities since evaluating the relations with gates and typical probability functions does not require much calculation power. It is easy to implement this method in tools. Solutions are not exactly precise, but since the probability functions are initially 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 assessment perspective, containing the most crucial elements only. Evaluation of the probability functions might be much less time consuming and expensive than testing the real system. Unfortunately, using probability functions created based upon guesses of the analysts might lead to wrong results. Testing the real system is much more reliable, but much more expensive, too.

Hence, in our methodology using Monte Carlo simulation for compositional risk analysis, we allow the analysts 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 real components while other parts of the system are only simulated using the probability functions and the directed graph of consequences modeled in the *Threat Composition Diagram*. The test results obtained from the real system components under test are then used to interpolate accurate probability functions and to improve the risk model. In our methodology, security testing of some component involves three steps: selection, test and update.

Selection. The first step is to select those components of the *Threat Composition Diagram* that should be tested using the real system components. Besides expert judgment how uncertain a guessed probability function might be, we suggest the following concept to identify the most crucial elements: For each incident *I*, two modified *Threat Composition Diagrams* are created. For the first, the probability function for the occurrence of incident *I* is set to null. For the second diagram, the probability function for the occurrence of incident *I* is replaced with a function that has a value of one if *I* is independent or if the incidents triggering *I* occur. For each modified *Threat Composition Diagram*, multiple Monte Carlo simulations are performed and the results which incidents occur within the same time periods are compared by evaluating the risk functions. That way, the impact of changes in the probability function for incident *I* can be measured. Those incidents for which the two complementary different probability functions result in the most different risk values should be tested in the first place since little errors in their probability functions will have the highest impact on the total risk picture.

Test. The second step is to test the selected elements using the real system. For actually testing how likely it is that some dependent incident will occur if some base incident occurs, the base incident needs to be generated. For independent incidents identified *vulnerabilities* and *threat scenarios* must be used to test if the incident might occur. Anyway test cases need to be created. It can be challenging, expensive and erroneous to manually create test cases. Instead of reinventing the wheel each and every time, it makes sense to use a catalogue of security test patterns [20]. Security test pattern do typically consist at least of a name, a context, a problem, a solution description and an expected result. We have defined a mapping between risk analysis elements in a CORAS *Threat Diagram* and test patterns in which the *threat scenario* is a direct counterpart to the problem description of a test pattern. Hence finding a fitting test pattern is easy if such a pattern already exists in a security test pattern database. The solution description of a test pattern typically contains some generic test procedure description that makes use of the *vulnerabilities* having relations leading to the *threat scenario* in the *threat diagram*. *Vulnerabilities* will be used as the input ports to pass test values to the system under test. The negation of the expected results in a test pattern correspond to the unwanted incidents that might be triggered if the *threat scenario* takes place. **Fig. 2** shows an example mapping. Generating test cases and executing them is manageable once a test pattern is identified and mapped to the risk analysis artifacts.



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

Fig. 2. Mapping *threat diagram* artefacts to test pattern

Update. The third step is to update the *Threat Composition Diagram* with the security testing results. Therefore, from the test results (i.e. for each test case incident *I* has been triggered or not), we interpolate a probability function P(T_1, T_2, T_S, L_X) for the occurrence of incident *I*. If we can find a good probability function describing the behaviour accurately, then we can use that probability functions in future Monte Carlo simulations for the entire system. This eventually gives us the opportunity to select other uncertain or critical components for real testing. If we cannot identify a sound probability function that gives a good approximation for the observed behavior, then we have to keep the real testing routine for that component in our Monte Carlo simulation process. Eventually later after having collected more test results, we will find a good approximation function. Even if we do not update the *Threat Composition Diagram*, we can continue with simulation and testing of other elements.

5 Conclusion, ongoing and future work

Though there are lots of technologies and tools for risk assessment and security testing, applying them for large complex systems is still a challenge. The integration of both together with the concepts of compositionality, Monta Carlo simulation and with patterns as shown here might help to reduce the effort and costs.

Development of the methodology and the tool described here is still in an early stage. We want to share our ideas that early to discuss them and to explore together with other interested researchers the full potential of compositional risk assessment with Monte Carlo simulation and security testing with test patterns as sketched here.

5.1 Case studies

Our efforts are driven by two case studies. These provide use cases and requirements inspiring our development. We plan to test and to evaluate our method and our tool by using them within these case studies. Additionally we will also analyze the same use cases with other existing methods and tools so that we can compare the results in relation to the effort for the different approaches.

The first case study is provided by a company developing solutions for eHealth systems. In the eHealth market sector there are lots of legal requirements concerning security and privacy especially for patient data. Risk assessment and security testing are currently done manually without much tool support and without a clear methodology. We try to improve this process and to measure the difference that our approach makes.

The second case study is about a large-scale trustworthy repository called the *S-Network* [19]. The *S-Network* is going to provide guarantees for the long term preservation and for the permanent secure non-repudiation accessibility of its content. Requiring all users to agree on a user contract, the *S-Network* will offer legal validity for its content, including verifiable metadata values (e. g. who stored what and when) with standardized legal implications for all participants. The *S-Network* is intended to become a universal platform for applications that have most stringent requirements, e.g. fair contract signing. Indeed, it must be resistant to both manipulation attempts and censorship. However, since it will not be possible to develop a perfectly secure solution, remaining risks have to be analyzed and communicated in order to create trust in the *S-Network*. Since the *S-Network* is designed to be a distributed long term archive having dynamic self-repair capacities, its risk analysis must deal with complex timing issues.

The eHealth case study is about a highly modular system that is already practically used and which will be gradually improved. It is going to show how our method can be applied for mature products. In contrast, the *S-Network* is currently only existing as a prototype. The further development of the *S-Network* is going to be driven by risk assessment and security testing. Thus, the second case study is going to show how our method can be applied throughout the entire development process of new systems.

5.2 Open Risk Assessment

Our vision is that risk assessment should become a process that typically takes place in an open collaboration. Risk analysis results should be made accessible for anybody as reusable artifacts. We plan to create a public open database for that purpose. This data would be helpful for other developers reusing the analyzed component as they could integrate the risk analysis artefacts in their own compositional risk assessment for their products. The end users could benefit from such a database, too, because they could inform themselves about the remaining risks in a standardized way.

6 References

1. International Organization for Standardization: ISO 31000 Risk management – Principles and guidelines (2009)
2. International Organization for Standardization: ISO Guide 73 Risk management – Vocabulary (2009)
3. Bouti, A., Kadi, D.A.: A state-of-the-art review of FMEA/FMECA. International Journal of Reliability, Quality and Safety Engineering 1, 515–543 (1994)
4. International Electrotechnical Commission: IEC 61025 Fault Tree Analysis (FTA) (1990)
5. International Electrotechnical Commission: IEC 60300-3-9 Dependability management – Part 3: Application guide – Section 9: Risk analysis of technological systems – Event Tree Analysis (ETA) (1995)
6. Lund, M. S., Solhaug, B., Stølen, K.: Model-Driven Risk Analysis – The CORAS Approach. Springer (2011)
7. Lund, M. S., Solhaug, B., Stølen, K.: Evolution in relation to risk and trust management. Computer 43(5), pp. 49–55, IEEE (2010)
8. Kaiser, B., Liggesmeyer, P., and Mäckel, O.: A new component concept for fault trees. In: 8th Australian workshop on Safety critical systems and software (SCS'03), pp. 37–46. Australian Computer Society (2003)
9. Papadoupoulos, Y., McDermid, J., Sasse, R., and Heiner, G.: Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. Reliability Engineering and System Safety, 71(3), pp. 229–247, Elsevier (2001)
10. Viehmann, J.: 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), DOI: 10.1109/SocialCom-PASSAT.2012.91
11. Erdogan, G., Li, Y., Runde, R. K., Seehusen, F., Stølen, K.: Conceptual Framework for the DIAMONDS Project, Oslo May (2012)
12. Erdogan, G., Seehusen, F., Stølen, K., Aagedal, J.: Assessing the usefulness of testing for validating the correctness of security risk models based on an industrial case study. Proc. International Workshop on Quantitative Aspects in Security Assurance (QASA'12), Pisa (2012)
13. Benet, A. F.: A risk driven approach to testing medical device software. In: Advances in Systems Safety, pp. 157–168. Springer (2011)
14. Kloos, J., Hussain, T., and Eschbach, R.: Risk-based testing of safety-critical embedded systems driven by fault tree analysis. In: Software Testing, Verication and Validation Workshops (ICSTW 2011), pp. 26–33. IEEE (2011)
15. Gleißner, W., Berger, T.: Auf nach Monte Carlo: Simulationsverfahren zur Risiko-Aggregation. RISKNEWS, 1: pp. 30–37. Wiley (2004), DOI: 10.1002/risk.200490005
16. Greenland, S., Sensitivity Analysis, Monte Carlo Risk Analysis, and Bayesian Uncertainty Assessment. Risk Analysis, 21: pp. 579–584. Wiley (2001)
17. Kolmogorov, A.: Grundbegriffe der Wahrscheinlichkeitsrechnung, Springer Berlin (1933)
18. Caprotti, O., Carlisle, D.: OpenMath and MathML: semantic markup for mathematics. Crossroads 6, 2 (November 1999), 11-14. ACM (1999) DOI: 10.1145/333104.333110
19. Viehmann, J.: The Theory of Creating Trust with a Set of Mistrust-Parties and its Exemplary Application for the S-Network, Proceedings of Tenth Annual Conference on Privacy, Security and Trust (PST 2012), pp. 185-194, IEEE (2012), DOI: 10.1109/PST.2012.6297939
20. Smith, B., Williams, L.: On the Effective Use of Security Test Patterns, Proceedings of the Sixth International Conference on Software Security and Reliability (SERE 2012) , pp. 108-117, IEEE (2012), DOI: 10.1109/SERE.2012.23