Bachelor thesis for a course of the
Anhalt University of Applied Sciences

Created at the Böblingen office of the
P3 Systems GmbH

Risk-driven Assessments for Security Functionality in Road Vehicle Information Systems

Submitted by Tilmann Matthaei
in February 2018
for achievement of the Bachelor of Science degree.

Matr. Nr.	4059918
Course	Applied Computer Science Digital Media and Games Development
Semester	7
Primary evaluator	Prof. Dr. Martin Kütz
Secondary evaluator	Prof. Dr. Michael Worzyk

CONTENTS

1 INTRODUCTION

Security assessments aim to identify vulnerabilities in information systems. Vulnerabilities might be the result of lacking data classification, bad security practices, bad architecture (design) choices, use of insecure protocols, limited monitoring capabilities, insecure third-party components [1] or even simple implementation mistakes. In contrast to the many potential vulnerabilities of the system, assuring the security of the system is often perceived as a stand-alone, downstream activity, where off-the-shelf solutions are applied to an otherwise insecure system. Security assessments are often limited to tests against this “layer” of security. Any penetration of this layer leads to complete compromise of the system’s functionality and information. Well-secured information systems rely on multiple layers of correctly implemented and verified functions.

The V model has been around since the 70s [2] and agile approaches like Extreme Programming (XP) have been around since the 80s [3], which have proven to be more effective in delivering high quality software. It is a distressing display, that even though these development methodologies have been around for so long, security assessments are still in a state where they are considered a final polishing and quality assurance task, as defined in the outdated waterfall model [4].

The lack of proper design, specification, verification and validation has led to a flood of insecure information systems being delivered. These systems require administrators to take additional precautions, as they have no confidence in them being able to protect themselves. Proper assessment of the information system, including its security design, can ensure that a system has appropriate security functionality at release. This makes the system more valuable to the customer, as he can expect a reduced risk level and maintenance costs.

1.1 SCOPE

This paper identifies methods to conduct development-accompanying assessments that judge the adequacy of information system security functionality, in particular for road vehicles. It shortly discusses adjacent activities and rationales of the stakeholders that are required to understand the intention behind the assessment activities.

This paper should enable a company, based on the risks associated with the operation of the information system, to:

- make a justified statement about the sufficiency and correctness of the security functionality and therefore the remaining risk.
- identify effective assessment cases that verify the sufficiency and correctness of the security functionality of the information system.
- adequately plan and execute a security assessment.

Methods described claim no compliance with local regulations, which are greatly varying between markets but instead seek to maximize the benefit for all stakeholders by producing an adequately secured information system. This paper also does not seek compliance with local or international standards, as many of them are criticized for their lack of applicability beyond “basic” functions, their low effort-to-benefit ratio, their inability to react to novel threats and their dependency from national authorities [5] [6].

Road Vehicle Specifics

At many occasions within this paper, statements and suggestions about security assessments are made that are not specifically referring to road vehicle information systems. This ensures comparability to related work and in most cases road vehicle information systems are not inherently different from other information systems. Additional information and recommendations, which are particular to road vehicle information systems, will be given within a “Road Vehicle Specifics” frame like this one.

1.2 Research Method

For this paper, an extensive literature review has been conducted. The literature used can be found in the appendix (cf. chapter 9.3). Based on the statements found, recommendations are derived by inductive reasoning. This method fits the problem, because it helps to deal with the complexity of the security assessment domain [7].

Inductive reasoning leads to probable conclusions, instead of secured ones. For example: “All planets ever observed were round, therefore probably all planets are round.” Even though the conclusion that all planets are round can only be proven, when all planets have been observed (which is impossible at this time), the statement is plausible.

In an economic context, relying on probable conclusions is often required to develop a business area. As only limited empiric evidence is available for some widely-accepted guidelines, some recommendations out of this paper might be invalidated in the future. This must be accepted as an inherent risk, when conducting a risk-driven security assessment as described in this paper.

1.3 Structure

Throughout this paper, an approach for security assessments is outlined. Chapter 2 clarifies basic terms and concepts of security assessments. Chapters 3 to 6 describe the elements of a security assessments in chronological order. In that regard, chapter 3 describes required resources to conduct such assessments, chapter 4 covers planning of contents and structure, chapter 5 gives additional recommendations for the execution and chapter 6 outlines typical artefacts and reporting processes.

As the described assessment focuses on risk reduction, in particular by tailoring the assessment around an initial risk-analysis, chapter 7 makes a comparison to other market-typical approaches. A summary of the findings and recommendations of the paper can be found in chapter 8. Recommendations for follow-up research are also given at that point. Closing this paper, the appendix chapter 9 contains definitions of technical term, an itemization of abbreviations used, the bibliography, and indexes of figures and tables.

2 Basics of Security Testing

This chapter gives an overview about the basic rationales and concepts of information security assessments. The information given shall serve as a baseline for the elaboration of a risk-driven approach to information security assessments from chapter 3 onwards. In addition to the definitions found within this chapter, explanations of technical terms used within this paper can be found in the annex in chapter 9.1.

2.1 Assessments and Tests

„Assessment” is a rather broad term used to describe “the act of judging or deciding the amount, value, quality, or importance of something, or the judgment or decision that is made” [8]. In the context of information systems an assessment is usually associated with a process that results in

a judgement about the ability of the system to meet or fulfil specified objectives. A usability assessment for example would be a process which results in a judgement if an information system is fitted for use through the customer.

Different terms are used interchangeably for assessments including terms like “evaluation” or “review”. All these have in common that it is not required but possible to have a predefined execution plan or a predefined way of reporting results. An assessment could also incorporate explorative methods like qualitative user or code reviews.

In the context of this paper, a stricter than usual distinction between tests and assessment is made. A test is always an assessment, but an assessment is not always a test. The reason being that a test requires a complete execution plan for the assessment process to be defined beforehand [9]. This obviously also includes the prerequisites, exit criteria, the way results get reported, etc.

Because of the prior planning, tests have the advantage, that they have a somewhat calculatable workload and they provide reproducible and comparable results. This results in them being significantly easier to incorporate in the system development plan and contract management.

The vocabulary from the following Table 1 will be used, because of the contradicting terms in use. The vocabulary was derived from the International Software Testing Qualifications Board (ISTQB) one [10].

Judgement about	*With or without predefined plan*	*With predefined plan*
multiple properties of the system.	Assessment	Test
a connected subset of the systems functionality.	Assessment suite	Test suite
a single property.	Case assessment	Test case

Table 1: Test and Assessment Terms
Description: The table shows the vocabulary for tests and assessments used in this paper.

Test cases are activities within a test to verify or quantify a single property, performance metric or suitability for a single application of an object. Usually, they are at least described by a precondition, the action to be performed, the expected behavior, and their result is expressed as the actual behavior and a resulting judgement. Table 2 gives an example of such a test case.

Summary	Ruleset Effectiveness - Ports
Preconditions	The firewall booted and is operational. Networks on both sides of the firewall exist through a network switch and a DHCP server.
Actions	Connect a PC via Ethernet to the “outside” switch. Connect a network sniffer to the “inside” switch. Apply ruleset “ruleset_T1” to the firewall. Generate packages for ports 1…200000 on the “outside” PC. Check for packages with the source IP of the “outside” PC in the sniffer logs. Repeat steps 3 to 5 with the rulesets “ruleset_T2”…10.´
Expected behavior	No package with a source IP from the “outside” network was recorded in the sniffer logs.
Actual behavior.	No package with a source IP from the “outside” network was recorded in the sniffer logs.
Result	Passed

Table 2: Example Test Case „Ruleset Effectiveness – Ports”
Description: The table gives an example for a test case, including the result, for a firewall. It includes a summarizing title for the test case, the preconditions to be met before executing the test case, the actions to be done by the tester, the system’s expected behavior, the system’s actual behavior in the test run and a result, based on the comparison between actual and expected behavior.

In contrast to a thorough planned test, there are also advantages, when incorporating explorative methods into an assessment [11]. These include that the most obvious (and therefore customer-visible) system defects will be discovered first and that design insufficiencies can be discovered.

2.1.1 Timeboxing

Timeboxing is a technique, where a maximum time of work on a subject is defined beforehand and work stops, if the time ran out, even when a desired result has not yet been achieved. The concept is not new, but it became increasingly popular in Software Development during the spread of agile methodologies. For example, in the agile development framework Scrum, all meetings are timeboxed [12].

Timeboxing can also be used to give more structure to assessments. When conducting a security assessment one might define an assessment case such as depicted in Table 3:

Summary	Timeboxed Network Sniffing
Precondition	A logger is installed as a man-in-the-middle in the Ethernet connection between the client and server. Client and server are running. Client and server are on their initial (startup state).
Action	The client is used for 30 minutes and the logger is collecting the traffic. A security expert tries to identify plain text information within the logged traffic for 30 minutes.
Expected behavior	No plain text information was identified.
Actual behavior	A non-encrypted HTTP request was transmitted and identified. The request exposed the user ID in plain text in a POST field.
Result	Failed

Table 3: Example Test Case "Timeboxed Network Sniffing"
Description: The table gives an example for an assessment case that has been time-boxed to 60 minutes total. The assessment case subsequently fits the structure of a test case.

With this method, it is possible to define an assessment case as a test case. Timeboxing therefore enables incorporation of explorative assessment methods into a well-planned test and makes results more comparable, effort more plannable, and bias more visible. Time-boxing can be used especially within the context of chapters 4.2 and 4.4 of this paper.

2.2 Confidentiality, Integrity and Availability

In the classic model of digital as well as physical information security, three objectives are considered. Confidentiality, integrity and availability of information should be maintained [13] [14]. In this context confidentiality means that information isn't revealed to an unauthorized third party, integrity means that information cannot be manipulated through an unauthorized third party and availability means that information is available to authorized parties. Some publications also include additional or different objectives such as nonrepudiation [15]. However, the classic set only contains the three mentioned above.

This model still works for systems that are only processing information on-demand or information systems that have physical components like road vehicle information systems. However, as often the concern is not the information itself, but the implications of incorrect system behavior following incorrect outputs of the system, it might be easier to define different objectives. For example, if the steering wheel sensor of a car produces incorrect outputs, which would be an impairment of information integrity, it might be possible that safety critical functions like ABS or ESP would not work correctly. The main concern here wouldn't be that the information integrity is impaired, but that safe vehicle operation is no longer possible, which should be considered the primary objective.

The difference in the understanding of the objective is one of the main reasons, why many information security frameworks are hardly applicable to on-demand or cyber-physical systems. In

this paper confidentiality, integrity and availability are therefore not considered as absolute objectives, but rather the risk-reduction and therefore reduced costs associated with them.

Road Vehicle Specifics

For road vehicles, the confidentiality, integrity and availability of stored data is not (yet) a concern in most cases. The most dangerous vulnerabilities are those that threaten the correct behavior (i.e. integrity and availability) of the electronic control units (ECUs) controlling the vehicle, and therefore the physical safety of the passengers. Any ECU or vehicle-connected infotainment system falls into the category of cyber-physical systems and the classic information security terms of confidentiality, integrity and availability are often not applicable.

2.3 RISK

"Quantitative risk analysis attempts to assign numeric values to the components of the risk (likelihood and potential damage) and to security controls (cost and effectiveness)." [16] From this quantitative understanding of risk we can model the cost $C(R)$ associated with a certain risk R by multiplying its probability of occurrence $P_r(R)$ with the financially rated loss $l(R)$ of the occurrence. $P_r(E)$ is the Bernoulli distribution, because either an attack targeting the vulnerability is attempted ($E = R = 1$) or it isn't ($E = \bar{R} = 0$).

$$C(R) = P_r(R) * l(R)$$

There are more in-depth frameworks for risk analysis like the USA's Department of Defense standard MIL-STD-882E [17] or the DREAD (proper name) method [18], but most are based on this simple calculation. Some express the loss in other (non-financial) terms. Nonetheless, the calculations below still apply, when risk is expressed in terms other than financial ones, such as casualties or abstract damage categories (high, medium, low).

On the other hand, there are costs associated with providing secure systems. These costs arise from efforts in planning, development, testing, implementation, documentation, hardware and third-party software costs among others.

Expenses for security functionality are investments [19]. If money gets invested into the project for security purposes the risks in the project decrease. As the risks of the project decrease, the overall balance of the system-producing project improves. Regarding a single risk this means that a security function has a certain effectiveness $P_f(F)$, which is the probability that the security function negates the negative impacts associated with the risk. $P(F)$ is the Bernoulli distribution. Therefore, the remaining, expected cost associated with the risk is reduced, as the attack would only cause a loss, if the security function fails.

$$C\left(R \cap \overline{F}\right) = P_r(R) * P_f\left(\overline{F}\right) * l(R)$$

It is acknowledged that this only holds true if R and F are independent events, which must be considered when estimating the probabilities of the events. This is especially important to consider, if the security function is impaired by previous exploitation of a vulnerability.

Multiple functions for the same risk have a multiplicative effectiveness, as secondary functions will only be demanded, if the first function failed to stop the attack.

$$C(R \cap \bigcap_{i=1}^{n} \overline{F_i}) = P_r(R) * \prod_{i=1}^{n} P_f\left(\overline{F_i}\right) * l(R)$$

The profit gained by a function $p(F)$ is the difference between the unmitigated risk and the mitigated risk.

$$p(F) = C(R) - C\left(R \cap \overline{F}\right)$$

For the net profit, the implementation cost of the function $C_i(F)$ needs to be deducted.

$$n(F) = p(F) - C_i(F)$$

The return on investment (ROI) R can be calculated from these values.

$$R = \frac{n(F)}{C_i(F)}$$

These calculations might seem complicated at first but they can be automated in a table calculation program. The following Table 4 shows an example calculation for the ROI of a single security function mitigating a single risk.

Description of risk	An attacker successfully gains access to the customer database through SQL injection.
Probability of risk	20%
Impact of risk	50.000€
Security function	User input is validated through an input framework integrated in the web application.
Cost of security function	5,000€
Effectiveness of security function	95%
Probability after function	1%
Risk level without function	10,000€
Risk level with function	500€
Net profit	4500€
ROI	90%

Table 4: Example ROI Calculation
Description: The table is an example of the calculated values from the previous formulas. There is a risk of a SQL injection (rated with 10,000€), which can be mitigated by a 5000€ security function, where input is checked before forwarding any values. As a ROI of 90% is very high, it would be advisable to implement the functionality.

Some aspects that should be taken into account when estimating the parameters are listed in the following Table 5.

Probability $P_r(R)$	*Loss* $l(R)$	*Effectiveness* $P_f(F)$	*Implementation cost* $C(i)$
Exposure	Downtime costs	Concealment	Development costs
Skill of potential attackers	Physical and mental harm	Increased skill and tool requirements for exploitation	Hardware costs
Benefit for the attacker	Loss of competitive advantage through the information	Reduced benefit for the attacker	Downtime costs
Required tools for exploitation	Loss of privacy	Logic impossibility of exploitation	Maintenance costs
Automatability	Reputation loss	Reliability of a third-party solution	Additional user effort
	Business disruption	Increased controllability	
	Additional user effort		
	Controllability		

Table 5: Aspects in Estimating Risks
Description: The table shows aspects of probability, loss, effectiveness and implementation cost, that should be considered when estimating their value. However, the table is not exhaustive and should be supplemented with aspects from personal experience.

2.4 SECURITY DESIGN

As the security needs of information systems differ based on factors like application, connectivity and system components, the distributor must tailor a security functionality for the information system. The theoretically constructed security functionality will be referred to as the security design (other common terms: security architecture, security specification) of the system. There are different methods to identify applicable security functions. They are usually separated from the specification of other functionality, because most system (non-security) functionality is determined by external requirements. Security functionality on the contrary, is determined by the risks that stem from the system itself. Security design is therefore a rather abstract, analytical process, which usually involves enriching a system design artefact like a system specification or UML (class) diagram with descriptions of risks and security functions. Additional security functions might also be considered from customer or assessment feedback, industry-specific or non-industry-specific security standards like the Common Criteria standard ISO/IEC 15408 [20].

Designing security functionality is usually the responsibility of a system or system security architect of the distributor, but responsibilities might be shifted towards the supplier, especially in an agile environment. A complete system design might not be available from the start of the project or is subjected to changes during the development process. This especially applies to system development projects which are conducted with an agile approach.

However the development is arranged, a clear and coherent security design helps implementation and project management to incorporate a fitting security functionality. Accordingly, a possibly changing, single security design artefact must be created (and possibly maintained), describing all security functions considered. This applies to functions which will not be implemented, because of cost considerations as described in chapter 3.1 or which have otherwise been dropped. This ensures proper tracking of security functions in later stages of the project to judge, if missing functions are a fault or have been left out justifiably. The security specification artefact can be a table or text document, a task list in a work organization tool like Jira, a Kanban board or a database among others.

This artefact will be the foundation for determining which assessment cases must be written and executed. This will be described in later chapters. It is required to peer-review this artefact or to get an independent assessment about the correctness and completeness of the artefact, because any

inconsistencies or missed opportunities in the architecture description might be costly and hard to fix in later development stages (cf. chapter 2.4.1). This assessment is described in chapter 4.2.

For larger projects, it must be considered if a dedicated framework for identifying and managing risks of the system should be implemented. Many of these frameworks are listed on the website of the British National Cyber Security Centre [21]. As risk analysis is the foundation for identifying effective security functionality (cf. chapter 2.3), proper design of the security functionality depends on the quality of it.

2.4.1 Security by Design

Security by design is a buzzword for describing that information security has been considered since the design stage in the development stage. This is the most cost effective approach to develop a secure system [22]. If the security functionality has never been specified, neither a security design inspection nor a functional security test can be conducted, because a security design artefact required for both. Subsequently, it would only be possible to conduct a system security assessment, where only common, known vulnerabilities are considered or to exploratively assess the system security. Both techniques cannot verify the sufficiency and correctness of the security functionality, as no knowledge about components worth protecting is available (sufficiency) and the implementation of security functions is only tested against input that is known to have caused faults in different systems, but not input that could be problematic in the context of this system (correctness).

However, many suppliers still follow an approach, where only few basic security functionalities mask the security faults of the system against common attacks. This might be a valid approach for low-risk systems, which are assumed to never be manually attacked by a somewhat competent attacker, but could be victim of automated untargeted attacks or a regular user who can identify obvious security faults.

Some of these suppliers still seek assertion from independent assessors for marketing or legal reasons, but limit the extent of assessment activities only to these, most common, attacks and security functions. Such request should not be conceded by a trustworthy assessor, because it puts his trustworthiness at risk and he might be hold partly liable for unidentified faults. If security is implemented in such a manner, automated test tools against common vulnerabilities usually only take few hours to set up and can be used by anyone, including the developers.

Therefore, security by design at the supplier's side is assumed to be a must-have for effective security assessment methods, including the one described in this paper.

Road Vehicle Specifics

Security by Design is still treated half-heartedly in the development of road vehicle information systems. Legitimately, the functional benefit for the vehicle is often focused in the development of the system. Historically, road vehicles were not connected (to the internet) and therefore had little need for IT security. This is changing and it is expected, that security concerns will be addressed more frequently during the design stage already, as security incidents and a growing demand for securely connected road vehicles will make it inevitable [63].

2.5 Sufficiency and Correctness

As described in ISO/IEC 15408 (Common Criteria), to defend the decision to accept the remaining risk of a product it is required, that the countermeasures against the threats towards the assets are demonstrably sufficient and correct [23](p. 24), which is also the legal regulation in most countries [24].

Unfortunately, it is complicated to identify if the security functionality is sufficient. In this approach, we define the sufficiency of security functionality by the circumstance that all (cost-)effective security functions have been implemented.

For this, two objectives need to be met:

1. All risks have been identified.
2. All security functions to mitigate the risk have been considered.

Risks and the security functions to mitigate them get identified in the process of risk analysis. Fortunately, a model of the system is usually created while planning the system. This model can be enriched with descriptions of risks associated with the different system components and corresponding security functions by the distributor.

However, the model might be erroneous or insufficient. In this approach, a security design assessment is performed to identify such architecture description faults. Additionally, the correctness (of implementation) of the functions is judged by the functional security test derived from the specification of the security functionality of the system.

Figure 1 shows how the different assessment activities verify the sufficiency and correctness of the system.

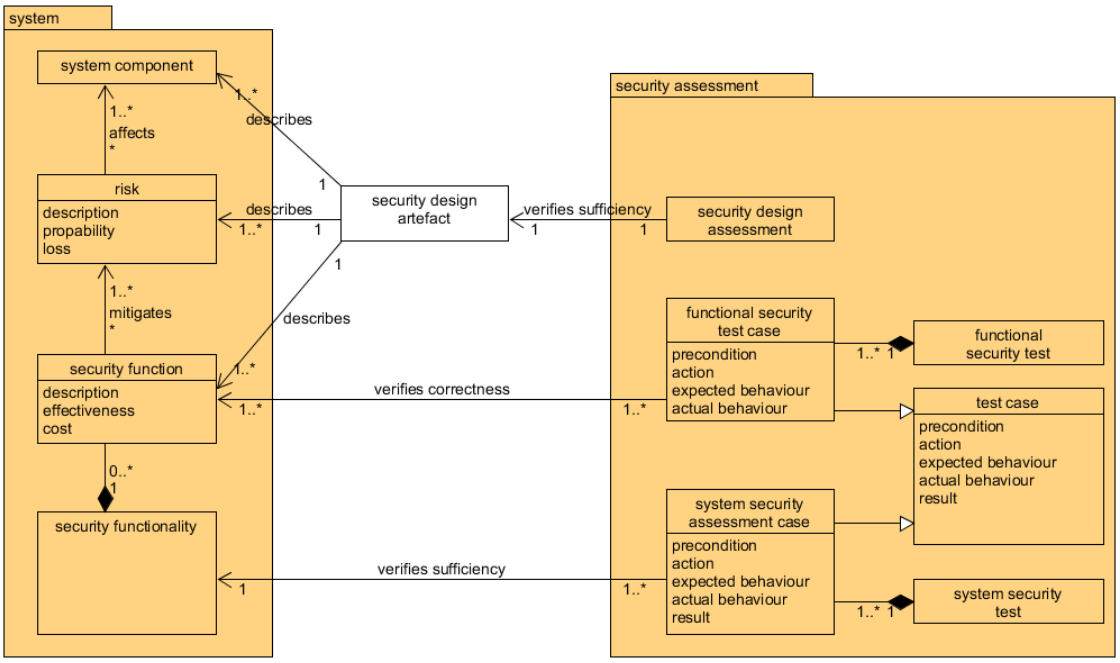

Figure 1: Model of Security Functionality and Assessment
Description: The diagram shows what security assessment components verify the sufficiency and correctness of the system. For example, functional security test cases verify the correctness of one or more security functions. The figure uses UML class diagram notation.

Obviously, there will be undetected risks remaining even though they are spare. It can be assumed that, because they have not been detected through extensive modelling and testing, their probability or their impact is very low and therefore their risk level is very low as well. Any additional security function would therefore only address insignificant risks and it can be concluded that the system is sufficiently secured (cf. chapter 2.5).

Road Vehicle Specifics

Road vehicles have many information processing components that are highly connected, use proprietary protocols and for which not many in-the-field attacks are known. Modelling a complete road vehicles' information system can therefore be an almost impossible task. On the practical side, ECUs are usually developed individually and the system boundaries can be assumed to be set at the bus they are connected to. It is the task of the vehicle manufacturer (customer) to define the trust status between the ECUs, while the unit manufacturer (supplier and manufacturer) only models the unit.

3 TEST RESOURCES

Usually, individual assessment suites are conducted for different system topics during development. The organizational structures that conduct these assessments are referred to as “assessment domains” in the following. To minimize the times in which staff, budget and tools are unproductive, it is advisable to share as many resources as possible with other assessment domains. This applies, if the resource sharing doesn’t bottleneck the security domain through competition for limited resources with other domains. Subsequently, the resource needs described in the following chapters should be understood as sufficient access to the resource instead of the resource being assigned to the domain. For example, if an assessment domain for Bluetooth exists, the security domain might not need an individual Bluetooth sniffer or, as a second example, the security assessment analyst could work part-time in the project and otherwise do specification inspections in other domains.

Consequently, the staff described in chapter 3.2.1 only exists as a role concept. Unfortunately, the skill set required for comprehending the complex interactions in which security faults often manifest is quite specific. For example, as a prerequisite for an ISTQB certification as a “Security tester”, “not less than three years of relevant academic, practical, or consulting experience” are required [25]. Consequently at least one full-time security assessment analyst should work (part-time) for the project.

3.1 BUDGETING SECURITY ASSESSMENTS

Spending money on the security of a product is often perceived as hard to justify. This is because security usually does not provide any user-experienceable features, but often is required by the regulations the organization must comply to. The business logic behind investing into the security of a system is that a loss is expected either way. Either it’s because of a successful attack on the system, or because of investments into the security of the system. The task is to find the point at which the expected loss is the lowest.

To get a clearer picture of the first possibility it’s advantageous to first have a look at historic data. The market volume (i.e. total revenue within the market) of the IT and communications sector in Germany in 2015 and 2016 combined was EUR 296.6 billion as reported by the German “Bitkom” association [26]. As this is the total industry revenue, it equals to the price of all products and services delivered by the German IT and communications sector within that time frame. The same association reported for the same time frame, that the German economy lost EUR 109.6 billion to data theft, industrial espionage and sabotage [27]. Even though a direct mapping of this number to the IT and communications market volume is not possible, because import, export and sector-internal sales would need to be accounted for. Nonetheless, it gives an idea of the security-related costs of typical products.

It is not a decisive factor if the customer must bear the losses through security incidents or if the liability is shifted to the distributor through applicable laws and regulations. If the liability is shifted to the distributor, the loss is directly affecting the distributor’s balance. If the liability stays with the customer, a product which is more secure has additional value to the customer and he is therefore willing to pay a higher price for it. However, as the customer might have limited understanding of information system security, most judicial areas feature liability laws for information security [24].

When deciding on the budget for the security of a system, one should rank functions by ROI (cf. chapter 2.3) and cut off the list of potential functions at a point, where the ROI is lower than what other investments (in different business areas) could achieve. The sum of the costs of the security functions is an appropriate budget.

As the most efficient security functions (with the highest ROI) usually get implemented first, further investments become increasingly inefficient which is illustrated in the following Figure 2.

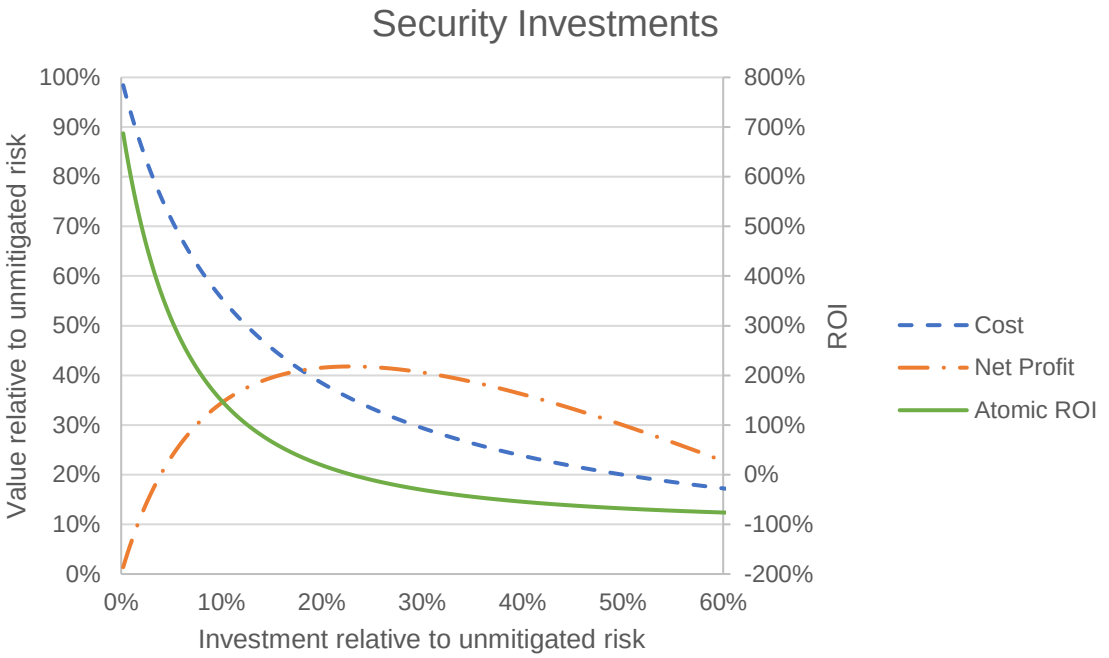

Figure 2: Security Investments
Description: The diagram shows how net profit, cost (of the risks), and atomic ROI (the ROI of the last security function implemented) change, when investing into the security functionality of the system. As the functions with the highest ROI are implemented first, the more is invested the less profitable the investment is. However, until the atomic ROI of the function reaches 0, the net profit still increases. The relative values chosen here are fictional and actually greatly depend on the system.

Testing plays a major role in the effectiveness of a function and therefore its ROI. In his blog, software development manager Jim Bird states that during design and development 5 faults get introduced into a software per “function point”. This equals to 50-55 lines of Java code of which 85% are caught before the software goes productive [28]. This means ~14 faults per 1000 lines of code (LOC) are present in an average productive system at release.

Other statistics have identified 15-50 faults per 1000 LOC as an industry average and 0.5 per 1000 LOC with independently tested software [29]. This shows that a system assessment is not optional, if high quality software shall be delivered. In the case of security functionality, it is required that bugs are as few as possible, because an attacker seeks out defect functions instead of working ones.

Yet, independent testing comes with a cost. It is estimated that testing consumes 25%-50% of software development time and budget [30](p. 2). When the development cost (without assessment) of a security function has been estimated within the security design, assessment should be accounted for. Preferably, this is calculated with historic data. If the distributor does not have historic data available, the safest approach from his perspective is to start with the assumption, that 50% of development budget will be consumed through testing and accordingly double the estimated development cost of the function. If the efforts are lower than expected, it is still possible to incorporate more security functions, which, even though they were initially under the ROI limit, may now be worthwhile.

Additional costs through assessment naturally change the ROI of the function and fewer functions are profitable. Untested functions however, are even less profitable, because of the high amounts of bugs that significantly lower their effectiveness or even decrease system stability.

To price an offer for a security assessment as an independent assessment provider or for internal cost tracking, it is a valid approach to upscale the cost of a small selection of assessment cases, representatively chosen from the security design inspection, security functions assessment, and the system security assessment, to the cost of the whole assessment. This assumes, that the costs

grow linear with the amount of assessment cases. Management, tool and others costs, which are not directly linked to the execution and reporting of test cases, must also be accounted for. It is even better to analyze historic data from other test projects and to scale the historic costs to the new project.

Road Vehicle Specifics

The losses conceivable for road vehicles are immense, as often the passenger's life is at risk. Even if the fault is detected early, vast recalls are to be expected, as most systems are only updated in the workshop. Therefore, proper risk analysis can reveal profitable investment opportunities in information security, where big losses can be avoided with little investment.

The use of proprietary protocols and system components that are easy to implement, but hard to test, can increase the security assessment costs share of the total development costs.

3.2 HUMAN RESOURCES

This chapter includes descriptions of relevant roles. The test-provider-internal roles will be described in chapter 3.2.1, while in chapter 3.2.2 external stakeholders will be characterized. The roles serve as a framework to manage the responsibilities and authorities within the assessment project. Roles help all participants of the project to identify their next task, as their scope of action is limited. This ultimately leads to more efficient work organization. Nonetheless, situations where a rigid role structure prevents required or beneficial actions should be identified and solved.

3.2.1 The Assessment Team

Some internal staff is required to ensure proper expertise and workforce to execute the security assessment. The role distribution follows the ISTQB model, which identifies five roles: test manager, test designer, test automator, test administrator and tester [30](p. 172 f.). However, the test automator and test administrator are not considered here, as they would have very limited duties within the security assessment. They can however, offer great benefit to the assessment activities if they are available to all domains by taking care of overarching, time-consuming tasks.

The *security assessment manager* is planning and administering all security assessment project resources and activities. He is the main contact point to all stakeholders for security assessment topics and steers the project in case of deviations from the plan. He only needs knowledge of system security in terms of adequate risk management and a general overview of national and international standards and regulations. His main skills are in the fields of business administration and customer relationships.

The *security assessment analyst* is the main source of information security knowledge for the assessment team. He gives recommendations for adequate assessment activities and equipment and can conduct explorative assessments such as the data transfer assessment described in chapter 4.4.4. As he relies on credible knowledge about system security, he must receive frequent, industry-rooted but independent training and time and material for individual learning. It can be advisable to deploy him part-time as a security tester to let him stay familiar with the challenges of the actual work. As there is a significant shortage in security talent [31] it is advisable to invest in training of this staff and to ensure staff satisfaction through good working conditions (including wages, management support and individual development options).

The *security tester* executes pre-defined test cases and records and documents system security misbehavior. He operates within a steady workflow, but needs a decent background in computer sciences to judge system behavior and operate with technical tools or frameworks.

Security assessments are often part of a system-wide assessment and are therefore integrated into a bigger assessment project. The *assessment manager* is the manager of this system-wide project. If the security assessment is conducted independently, usually the security assessment manager is in charge.

The *human resource management* supports the security assessment by employing and training suitable personnel.

3.2.2 External Roles

Information system development is an effort in which multiple stakeholders are involved. In this chapter these external roles are described. In reality, these roles can be shared or coinciding, even among multiple businesses or in one person. Monetary flows described might have a virtual nature within a company.

The *supplier* (company) is commissioned by the distributor to develop (i.e. design, implement and possibly maintain) a system based on a specification. The correct development of the security functionality is the supplier's responsibility. He is liable for faults in security functionality committed to deliver.

The *distributor* (company) produces the system and sells it to customers. He can charge a premium for secure software from the customer or can alternatively reduce costs associated with liability claims by delivering more secure products. Specifying the correct security functionality is the distributor's task, even though he might need supplier or independent consulting to do so. His main tasks are related to market research, marketing and financing of information systems.

The *customer* (company) purchases the system and expresses the need for a system and its security functionality through a commission or market behavior. It is possible, but not necessary, that the customer and user are the same person. Where this is not the case, often liability for user harm caused by lacking security functionality is shifted to the customer (and possibly from there to the distributor) by local regulations.

The *user* is the person that uses the system and often is the person that is affected by the direct damage associated with a successful security breach of the system. He might be negatively impacted by a loss of data, remediation costs or physical and mental harm. Especially in cases where he uses heavy machinery such as a car, plane, oven or construction machinery, consequences can be fatal.

In the context of this paper, the assessment team verifies that the supplier's system matches the expectation of the distributor. Additionally, the assessment team supports the development process by identifying improvement potentials of the security functionality. A common contractual structure is the commission of the supplier for the delivery of the system and an additional commission of the assessment by the distributor for the assessment of the provided system releases. This contractual structure might relate to permanent employment of the assessment team through the distributor or a commission of an independent service provider. An overview of this structure is shown in Figure 3.

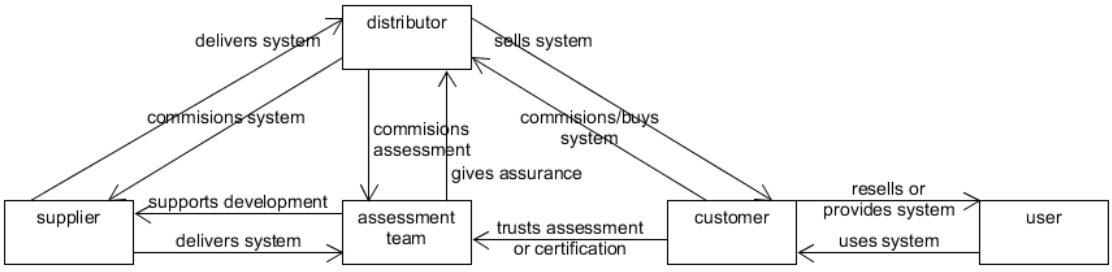

Figure 3: External Roles
Description: The figure displays the relations between the stakeholders of a security assessment. This structure applies, if the assessment team has been commissioned by the distributor to conduct a security assessment for a system delivered by the supplier. The organizational separation must not necessarily relate to a legal separation of the entities.

Another option is a commission of the assessment team by the customer. A positive assessment result would then be an acceptance criterion for the system delivery. The supplier might also commission an assessment as an internal quality assurance measure.

3.3 ASSESSMENT ENVIRONMENT

Systems might require a certain set of resources and signals to turn into an operational state. The most vital one is a power supply, but some systems might have additional requirements. These can be wake-up or keep-alive signals over an external interface or a bus master to manage data transfer. A system might not even process any data if no apt server is available. Fulfilling these additional requirements helps to make the assessment more comparable and more realistic in regards to the expected use of the system.

Many of these preconditions are constant between assessments and can therefore be outsourced from the assessment case's preconditions to the assessment environment. This environment should represent the environment in which the system will operate after release. It is required that the environment is documented in case of later queries, as a system's behavior can depend on the environment it operates in.

In security assessments anomalies in the environment are often created to assess system behavior. Security faults regularly arise when the logic, which is assumed to be firmly laced into the environment, is invalidated by chance or an attacker. Security assessments must therefore consider the assessment environment as something "fluid", where too little environment logic leaves the system invulnerable, because it is not operating correctly. Too much environment logic might mask security faults which an attacker could exploit, if he can change the environment around the system. In the context of security assessments, one should therefore think of the environment as the minimum operational state of the system at which it processes information that is worth securing and mention any other system dependencies in the preconditions.

Cyber-physical systems are software and hardware systems that interact with external sensors and actuators and through them with their environment or that act as a master for such systems. Their behavior can be greatly influenced by the signals they receive from their sensors and actuators and therefore realistic feedback to the signals they send is required. The class of cyber-physical systems consists of ECUs, supervisory, control and data acquisition (SCADA) systems, diagnostic and operation systems and most embedded systems among others.

Realistic sensor data can be obtained by mounting the system in an actual but isolated target environment. This environment could be a test vehicle, test production line or other proper physical environment. On the contrary, even though assessing the security of a system in such an environment is desirable, for some assessments, mounting the system into it is too expensive or dangerous. For example, testing the behavior of an ECU for brakes at the maximum speed value of 255 km/h in an actual vehicle puts the tester at great risk. Therefore, a sophisticated test bench should supply accurate but manipulatable sensor data through a simulated environment. This means in practice that the system to be assessed has all external interfaces connected to a simulation system's external interfaces (for example a PC running a bus simulation). It is desirable that manipulated data can be send directly from the simulation system (for example manipulated TCP/IP packages), so the same system can be used for simulation and input generation.

Hybrid solutions as shown in Figure 4, where isolated actual environment pieces are connected to a simulated environment, are also feasible and can help to more realistically test interactions with single external systems.

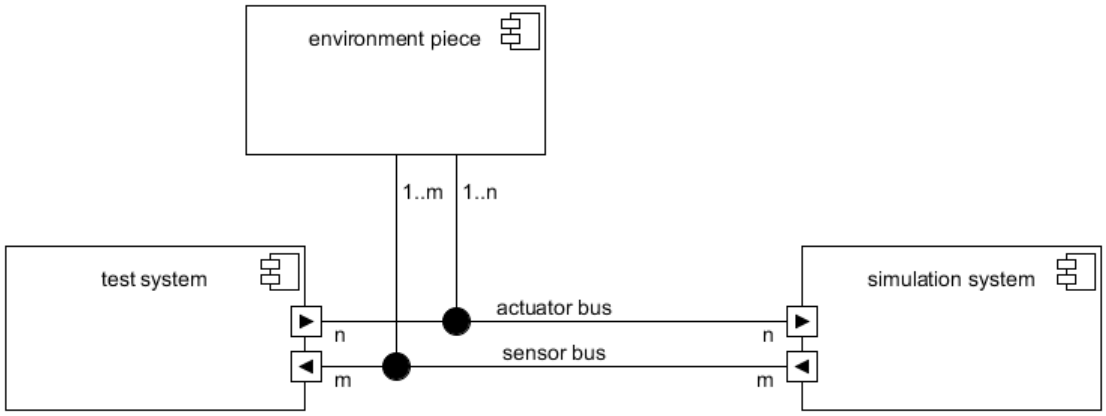

Figure 4: Hybrid Test Bench
Description: The figure depicts how the system under assessment is connected to both a physical environment piece and a virtual representation of one or more other environment pieces inside a simulation system. The actuator and sensor busses are displayed independently here, but might in reality use the same cable.

Road Vehicle Specifics

Road vehicle information systems are cyber-physical systems. A, usually 12V, power supply is required to operate the system. Additionally, it is often dependent on sufficient vehicle-internal bus traffic (CAN, MOST, FlexRay, internal Ethernet depending on the vehicle) and sufficient and realistic bus traffic should therefore be created to induce a realistic system state. This can be done by using an ECU simulation framework.

3.4 Measurement and Logging Equipment

To ensure that the developers can fix security faults as soon as possible, the system input, system output and system state should be logged. This requires loggers for the interfaces relevant for the assessment case, as well as a logger on the system that captures the system state (which is often referred to as debug tools). These logs should be attached to each defect together with the actual measurement values from the test to aid the fixing process (cf. chapter 5.1).

For each interface that the system offers (and that is under test) tools are required that can receive and transmit manipulated data in all relevant OSI layers. For Ethernet, this would mean that both an Ethernet sniffer (like Wireshark, potentially with low-level sniffing hardware) and a generic Ethernet transmitter, which is available in hardware in most PCs, is required.

Especially for transmitting data, programming on a driver like WinPcap might be required. The effort and resource requirements for these kinds of developments are relatively high. Therefore, reusable tools are preferable and give a big market advantage, as their development costs only occur once. It is beneficial if the measurement tools are integrated in the simulation system, as dependency issues can be more easily assessed by manipulating otherwise valid data traffic.

Road Vehicle Specifics

Logging should be available for vehicle-internal busses (CAN, MOST, FlexRay, internal Ethernet). System traces might be collectable via a network log protocol. For mobile device connectivity, often WLAN, Bluetooth and USB are available and the traffic on these interfaces should be logged, when they are being assessed.

Logging and simulation combinations are available for vehicle-internal busses. Other interface input frameworks might have to be programmed. Sniffers for WLAN, Bluetooth and USB exist and can be used to capture the system output.

3.5 ORGANIZATIONAL RESOURCES

Within the limited scientific and programmatic publications, often a process of universal information and a subsequent independence of workers from management structures is described [32] [33]. In the following, stakeholder information paths are described non-exhaustively.

3.5.1 Work Organization Tools

It is possible to track system components, risks, security functions and test cases in individual tables. However, because of the many 1-to-n and n-to-n relations it is advisable to keep these models in more appropriate formats. In hypertext format, it is possible to encode relations into links, while ticketing solution like Atlassin's Jira or Microsoft's Team Foundation Server, can encode the relations as ticket relations. This enables the developers and the testers to more efficiently explore the security functionality of the system.

Additionally, all works results should be trackable within such a system, so the progress is linked to components, risks and functions and reporters don't need to switch between multiple tools. The tool must implement an effective role system and work history for each element to ensure the confidence in the data for any reports.

Road Vehicle Specifics

As Information systems for road vehicles are often developed by a third party, work organization tools should support distinguished role and permission management.

3.5.2 Training

Employees should be given enough possibilities to receive security-related training if they conduct information security assessments. At this point, no guideline for a complete training framework will be given, but it is acknowledged that a holistic training procedure is a necessity in today's work environment [34]. In addition to the general training contents, members of the security assessment team should receive the information sources and trainings described in the following.

The whole assessment team should receive information through the following avenues:

- Access to relevant information security magazines to keep up-to-date with current security trends.
- Mandatory orientation training / mentoring for the security assessment process to enable the employee to promptly work effectively in the company / project.
- Mandatory confidentiality and security ethics training to minimize sensitive information leaks to third parties. This also supports that all activities are within the regulatory and ethical boundaries of security assessments.
- Training about the security issues for the specific industry, so that faults can be properly found and judged.
- Training about the overall most common security issues, so that faults can be properly found and judged.
- Training for adopted equipment and environment to enable everyone to properly handle it.

Additionally, security assessment engineers should receive information about the following topics, to improve their information security knowledge:

- Access to relevant standards to provide best-practice and certification information to the supplier and to use them for the assessment strategy (cf. chapter 4.1).
- Access to relevant research publications and statistics to enable justified decisions about assessment techniques, risks and functionality.
- Travel possibilities to security trade fairs to adopt new equipment, environment pieces, services and practices for the assessments.

In addition to the basic training, that the whole assessment team should receive, security assessment managers have additional knowledge needs. Therefore, the following matters should be covered for them:

- Access to relevant standards to adopt new workflows and best-practices.
- Mandatory training on confidential workflows and information forwarding to plan tools and processes where sensitive information will not leak to third parties.
- Business intelligence training to enable justified, data-supported management decisions.

It is the responsibility of the security assessment engineer to communicate personnel needs to the human resource management, including the training contents. As personnel processes are rather slow, personnel needs should be communicated early.

Road Vehicle Specifics

A training about ISO 26262 [62] for the security assessment engineer and security assessment manager is advisable, because it is one of the most applicable standards. Everyone should receive a basic training about the strengths and weaknesses of the bus technologies in use (CAN, FlexRay, MOST, LIN, etc.), the typical ECUs and typical interfaces. This helps to ensure that everyone hast the ability to properly assess the security functionality.

4 ASSESSMENT PLANNING

An assessment plan should describe the strategy and contents of the assessment. The contents of the assessment are the assessment cases (cf. chapter 2.1). They can be identified for the different stages of security assessments described in chapters 4.2 to 4.4. These chapters give recommendations about how the assessment cases should be designed.

In addition, an overarching strategy should give order and context to the assessment cases. The strategy aims to ensure a punctual start of all assessment activities and a holistic assessment of the overall system. The process of determining such a strategy is described in the following chapter 4.1. The assessment plan is oriented towards the “test plan” definition of the ISTQB [10].

4.1 IDENTIFYING A SECURITY ASSESSMENT STRATEGY

A strategy is “a detailed plan for achieving success” [35]. Consequently, it is first required to understand, what a “successful” security assessment is.

It is a conclusive model that defects get more expensive the longer they stay in the system, as their interactions with other system parts increase. Compared to the design stage defects are 6.5 times more expensive to fix during implementation, 15 times more expensive during release test and 100 times more expensive to fix during maintenance [36].

A successful security assessment is therefore one that can (within a reasonable budget), identify security faults as early and completely as possible. A proper assessment strategy should “clarify the major tasks and challenges of the test project” [37]. The security assessment strategy should therefore cover the entry criteria for planned activities, the assessment techniques to use and the depth of assessment, an estimation of resource expenditure throughout the project and solutions for common problems (e.g. budget shortages or premature project closure).

Together with the relations between assessment activities and supplier deliveries identified in chapter 2.4, a rather abstract assessment strategy can already be established. The security design must be validated with a security design inspection, the security functions need to be verified by the functional security test and the final system’s security posture is asessed by the system security asessment. The activities start once the supplier delivers the object under test and once the previous activity finished. Faults found are reported immediately to the supplier and a final report

summarizes all test results. However, a tailored schedule based on the development schedule should be created. This tailored schedule must take the development lifecycle into consideration, because of the varying availability of development artefacts in different development lifecycles (cf. chapter 4.1.1).

Appropriate assessment cases must be designed based on the methods described in chapters 4.2 through 4.4. They should be peer-reviewed to reduce the subjectivity of the expected fault detection quota and subsequent poor assessment case choices.

Road Vehicle Specifics

Road vehicle information systems can be complex and have long development times. Therefore, it is advisable to find small subsets of function assessment cases that are linked and execute them together. The system can then be tested against typical vulnerabilities of completely integrated components and a complete system security assessment can be executed, when only minor bugs are to be fixed by the developers. Delays, because of a late start of security assessment activities, are expensive. If the assessment cannot be finished in a timely manner, there might be a tradeoff between exhaustive assessment and early release, when deciding when to start the system security assessment.

4.1.1 Matching Security Assessment Strategy and Development Lifecycle

The security assessment process starts with the definition of a security design inspection procedure. Before the inspection of the security design artefact, very little is known about the security needs of the system. Succeeding the acceptance of the overall system design, the development starts at the supplier side.

At this point, the security assessment analyst can define the test environment in agreement with the distributor and general resources (such as rooms, staff capacity or office equipment) can be acquired. In this stage a project plan and security assessment strategy, should be defined by the security test manager based on the system development plan. In cases where a complete system architecture description and/or system development plan is not available, which it sometimes isn't in agile projects, some assumptions about the system must be made. If these assumptions later prove to be wrong, there is the possibility that resources have been acquired which will not be needed or that test plans cannot be uphold. This creates additional costs, which is a problem inherent in agile development methodologies and a trade-off for other benefits associated with agile methodologies.

Once a security function has been agreed to be implemented and specified, a security test engineer can identify the test cases to be conducted to verify the function during the functional security test with the methods described in chapter 4.3. An imprecise or inconclusive security design artefact might make it impossible to define a test case [30](p. 22) in which case this information should be feedbacked to the supplier. Lacking documentation can cause functional or security faults.

The information about security functions designed must be delivered by the distributor. This can be an iterative process. An appropriate form of communication about new or changed specification must be introduced with the distributor. When the developers have implemented one or more functions to a level where they assume them to be "done", the supplier must deliver a system snapshot to the security testers and communicate the availability of new functions.

The security testers can then execute appropriate test cases and should feedback information about found faults to the supplier as soon as possible, so the developers can work on fixes. Implemented fixes should then be communicated again to the testers so the test cases can be executed again to verify the fix has been effective. In this stage, testers should document their results and collect logs and test results into a centralized tracking system (cf. chapter 5). Most software today is implemented in a modular manner, where system components work somehow independently from each other. If it's possible to create a functioning system snapshot from these components, test

cases can be conducted on regularly created snapshots. Otherwise if unit tests are not possible or not useful to judge the security of the system, only a release security test is feasible.

A system security assessment should be conducted to identify issues in specification and implementation that have not been detected previously. Additional assessment cases for this should be written by the security assessment analyst once the security functionality has been agreed on.

After all assessment cases have been executed or a reportable subset of assessment cases have been executed, the security assessment manager should create a summarizing report and handle project or subset closing. If permanent maintenance for the system is planned, a release report should be created by the security assessment manager and additional maintenance test activities might have to be planned. The system should then be integrated into the security management cycle of the customer.

To streamline communication, it might be useful to get and provide distributor information through a single channel for all assessment domains. Information about the implementation and test status of protected system components might also need to be communicated, if many assessment cases depend on the system component functioning correctly (for example in fuzz testing). This can likely be done within the assessment organization between the assessment domains to unburden the distributor.

An overview of the general assessment strategy is shown in the following Figure 5.

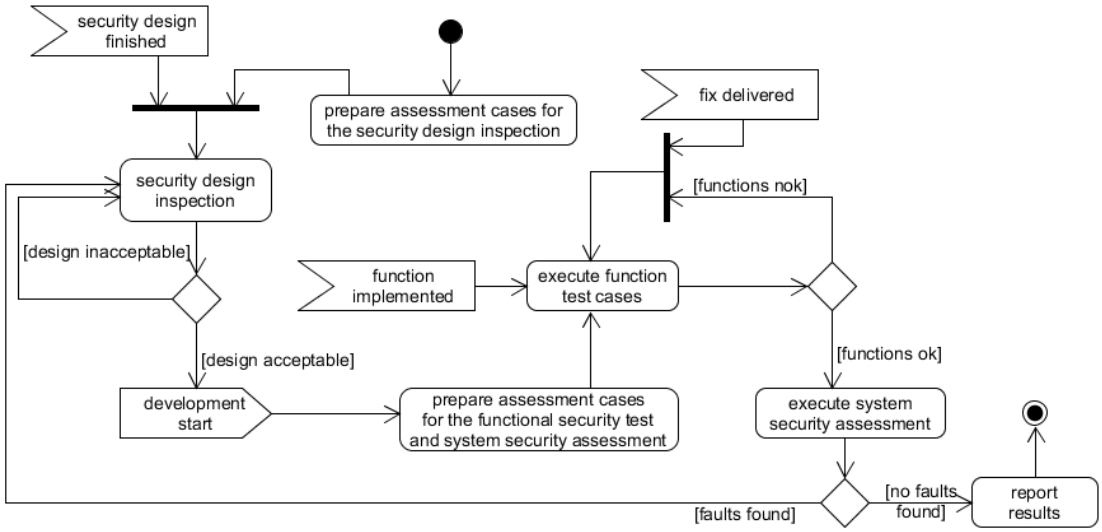

Figure 5: Security Assessment Strategy
Description: This UML activity diagram gives an overview about the security assessment process. For simplicity, the function test cases are displayed cumulatively. In reality, the information that a function has been implemented would trigger a single test case execution. The possibility of development closure is not considered here as well for improved readability. This would be the only case in which the final report should disclose any faults.

4.2 SECURITY DESIGN INSPECTION

The security design artefact should model system components, risks and considered security functions. These are often in 1-to-n and n-to-n relations, which makes modelling them without hypertext quite challenging and prone to mistakes. A system component may have multiple risks associated with it. A risk could be mitigated by a whole bunch of security functions or none and one risk might impact different system components.

The set of system components is assumed to be correct in the context of the security design, because the general system specification is created independently from the security design artefact

and components should follow the general system specification. It is useful to remember that system specification might change over time through formal change requests or within agile structures. Therefore, the security design artefact should be updated based on changes of the general system specification.

The security design inspection aims to verify sufficient security functionality has been specified for the system and should answer the questions:

- Are estimation methods chosen correctly and coherently?
- Have all risks associated for each component been correctly identified?
- Are these risks correctly quantified in terms of financial loss and probability?
- Is this risk applicable and limited to the identified system components?
- Have all security functions for each risk been correctly identified?
- Are these security functions correctly quantified in terms of cost and effectiveness?
- Are the security functions applicable and limited to the specified risks?

At this point, it must be recognized that the security design artefact is created through many approaches, in many forms and from very different perspectives. A bank might be able to express all risks through a financial value that can be calculated from their transaction volumes, while a manufacturer of a road vehicle's biggest concern might be impairment of safety critical systems. The distributer might also use different modelling techniques like decision trees [38] a dedicated threat modelling tool/framework, a general (linkable) ticketing system like Atlassin Jira or Microsoft Team Foundation Server or a table or database tool. Within a project, a singular solution must be chosen, based on the project needs to enable proper assessment and implementation of the design.

The broad variance of methods and representations used makes the models prone to inconsistencies. For example, it would be possible to consider high risk tolerance of the customer in two ways: As a higher ROI limit, because the customer is only willing to pay for the most effective security functions or as a reduced expected loss, because the customer would buy the system even though it has lacking security functionality. Both are equally valid approaches, but if both are applied to one security design artefact, potentially implicitly hidden in a framework, too few security functions will be implemented, as the spending willingness of the customer has been underestimated. It could also be possible that some mechanics have not been considered at all. For example, a distributor might consider costs only in terms of liability claims, but has not considered that a customer might pay more for a more secure product or that access to security critical markets, such as the finance sector, can be lost. The assessor should check for such model inconsistencies first, before considering if the system has been adequately described within this model.

It should also be kept in mind that the vocabulary of other methods might be different. For example, risks could be called threats and security functions countermeasures [39] or connection components might not be identified as system components [40], even though they could impose risks as well. The assessor would need to map this vocabulary to the one used in this paper.

What kind of representation of the security design artefact has been chosen does not matter. The only concern is, if all components, risks and security functions have been adequately considered. To identify inconsistencies, it is advisable to follow a bottom-up approach, where the assessor starts on the component level, then examines the risks and the associated security functions last. As risks originate from the components and security functions originate from the risks they should mitigate, a bottom-up approach ensures that parts of the security design do not have to be reassessed, because of a security design fault found later in the inspection.

The actual inspection method can be chosen somewhat flexibly and should represent the overall security expectation for the system. Inspection methods range from pragmatic criteria catalogues [41] to moderated discussions [42]. In general, the higher the security expectation for the information system, the more people should be involved and the more aspects should be inspected. Inspections that rely on interviewing the designer offer the additional benefit, that the practice of

explaining his thought process and having it criticized helps the designer to improve his modelling and design techniques as well.

4.3 FUNCTIONAL SECURITY TEST

Functional security test cases verify, that the implementation of the security function matches the security design. They should be written by a security assessment analyst and peer-reviewed.

The following should be validated for each security function:

- Does the security function stop undesirable use of the component it protects to the extent specified?
- Does the security function allow desirable use of the component it protects to the extent specified?

The easiest way to find functional test cases that can be executed with reasonable effort, but covers most faults, is equivalence partitioning. For equivalence partitioning the set of inputs is divided into subsets, that are likely going to produce the same failure. In non-security functional testing, usually the equivalence classes are identified within two types of equivalence classes, invalid input equivalence classes and valid input equivalence classes [3].

For security testing, it is useful to type the equivalence classes into valid, unauthorized and invalid ones. Valid input equivalence classes cover inputs that are correct from a protocol, application logic and security standpoint. Unauthorized input equivalence classes cover inputs, where the input is correct from a protocol or application-logic perspective, but access is restricted within the current authorization context. Invalid input equivalence classes cover inputs, where the input is invalid from a protocol or application-logic perspective, regardless of the authenticity of the input. It is easier to identify all the possibilities for inputs with this structured approach. Table 6 shows equivalence class partitioning on the example of HTTP authentication.

Input Condition	*Valid equivalence classes*	*Unauthorized equivalence classes*	*Invalid equivalence classes*
User / Password	Valid user, valid password	Invalid user, valid password	No user specified
	Valid user, no password	Valid user, invalid password	
		Invalid user, invalid password	
Authentication field	One authentication field: Valid user, valid password, supported authentication method	No authentication field	Multiple authentication fields
	Authentication field on unrestricted content	Unsupported authentication method defined	No authentication method defined
			No separator ":" encoded
			Multiple authentication methods defined

Table 6: Equivalence Partitioning Example HTTP
Description: The table displays equivalence class partitioning with three types specific to security testing on the example of HTTP authorization through the basic authentication scheme described in RFC1945 [43]. Clients can authenticate through this scheme by appending a header field with the format:
Authorization: Basic base64(username:password)

where base64(username:password) is a base64-encoded string of the username, a ":" separator, and the user's password.

From an equivalence class, a test case is derived by choosing a representative input and then specifying preconditions, actions and expected behavior for supplying it to the system. Where applicable, boundary values should be chosen as inputs, as errors are most likely to occur for boundary value inputs [3]. Table 7 and Table 8 each give an example for a test case derived from the equivalence classes of the examples in Table 6.

Summary	User / Password: Valid user, valid input
Precondition	A HTML file has been configured to only be accessible through basic authentication on the web server. A user "tester_0001" with the password "test123!" Has been configured on the server and permitted to read the HTML file.
Action	Send a HTTP request for the file to the server with the header field: Authorization: Basic dGVzdGVyXzAwMDE6dGVzdDEyMyE=
Expected behavior	The server responds with a HTTP response and delivers the HTML file in the message body.

Table 7: Example Valid Equivalence Class Test Case
Description: This test case verifies the inputs from the "valid user, valid input" equivalence class from Table 6 are handled correctly.

Summary	User / Password: Valid user, invalid password
Precondition	A HTML file has been configured to only be accessible through basic authentication on the web server. A user "tester_0001" with the password "test123!" Has been configured on the server and permitted to read the HTML file.
Action	Send a HTTP request for the file to the server with the header field: Authorization: Basic dGVzdGVyXzAwMDE6dGVzdDEyMw==
Expected behavior	The server sends a 403 (forbidden) response with no message body.

Table 8: Example Invalid Test Case
Description: This test case verifies the inputs from the "valid user, invalid password" equivalence class from Table 6 are handled correctly.

For every function, at least one test case should exist, to ensure basic availability of all functions. In most cases, equivalence class partitioning reveals more useful test cases to be conducted for the function.

In addition to the functional test cases, it might be useful to have multiple test cases for different quality attributes of the security function. Quality attributes are the overall factors that affect run-time behavior, system design, and user experience and have often been considered in the specification of the function already. Quality attributes at run-time are mainly availability, interoperability, manageability, performance, reliability, scalability and security [44]. Security function security will not be considered for now, but is assumed to be covered through the correctness of implementation.

Examining different quality attributes might reveal different ways in which security functionality can be impaired. For example, availability of a firewall is required to access the network behind it, bad performance of a user authentication function might make the system vulnerable to DoS attacks or lacking manageability could increase response times to a security incident if certificates need to be revoked.

The quality of the function should reflect the expectations of its effectiveness and cost expressed in the risk analysis. This especially applies to possible downtime and maintenance costs through poor availability, manageability and reliability as well as limited applicability and therefore effectiveness, because of lacking interoperability and scalability.

Function tests can usually be classified into black-box tests, white-box tests and analytical tests. Which method is the most appropriate for testing should be decided on a per-equivalence-class base. Black-box tests are appropriate for all integrated functions. White-box tests can be used additionally for functions, that are so critical that misbehavior should be remediated, even if it doesn't propagate to the outside or where only late integration is possible. Analytical testing can be easily automated and can support system development early, even when components are not fully developed yet.

Road Vehicle Specifics

Because of the high complexity of road vehicle information systems, the test focus might be shifted from function tests to higher level test and only basic verification of the security functions is performed. This might be a valid approach, especially if exhaustive higher level input tests are performed.

4.3.1 Black-Box Tests

Black-box tests are tests where no code and interface specification is available to the tester and run-time tests are derived from them.

Test cases are then only defined by their input and output on one or more IO ports, including the HMI. Input is transmitted via a signal generator (e.g. a network card) and the response is received through some measurement equipment (e.g. system logger, network sniffer and screen capture).

Black-box tests are especially efficient, when the goal is to find system defects that propagate to the outside and therefore might expose a security vulnerability to an attacker. Because of the slow data exchange on the IO ports compared to the memory and the limited knowledge about the data processing, black-box tests struggle to identify failures, that only occur if a very specific set of inputs is transmitted or that require inputs on multiple interfaces at once. Fortunately, attackers will be at the same disadvantage and therefore these vulnerabilities are less likely to be discovered after release as well.

Road Vehicle Specifics

Because many proprietary protocols and components are used in road vehicle information systems, often black-box tests are a significant part of the functional security test.

4.3.2 White-Box Tests

White-box tests are tests where any code and specification is available to the tester and run-time tests (i.e. tests where the system is in operation) are derived from them.

For software white-box testing, a popular approach is to write actual code test cases using a test library (and possibly framework) like JUnit (Java), xUnit (C#, .NET) or gtest (C++). Results can then be automatically collected and summarized, often within the development environment.

The following example shows how, in such white-box testing frameworks (in this case gtest), tests can be expressed with their preconditions, inputs and expected outputs.

In the following example the class "User" is under test. It has an overloaded setUsername() method, with which the username can be stored and a getUsername() method to retrieve it. The setUsername() method shall be tested for proper handling of null pointer parameters.

```
class User {
public:
 User();
 virtual ~User();
 void setUsername(std::string*) throw (NULL_EXCEPTION);
 void setUsername(std::nullptr_t) throw (NULL_EXCEPTION);
```

```
  std::string getUsername();
private:
  std::string Username;
};
```

At first the preconditions can be defined in gtest's Test class. In this case, no separate preconditions are required.

```
class userTest : public ::testing::Test{
  public:
        void SetUp(){

        }
        void TearDown(){

        }
};
```

Type-casted and non-type-casted, null pointers are given as a parameter to the setUsername() method within the gtest TEST_F macro, which is used for tests with a fixture (i.e. a Test object). The expected output is that the class throws a custom-defined NULL_EXCEPTION, instead of overwriting the username pointer with the null pointer or any other undefined behavior.

```
TEST_F(userTest, usernameNull){
  std::string *test_username = (std::string*)nullptr;
  User test_user;
  EXPECT_THROW(test_user.setUsername(nullptr), NULL_EXCEPTION);
  EXPECT_THROW(test_user.setUsername(test_username), NULL_EXCEPTION);
}
```

With the Eclipse C/C++ Unit Testing Support package, a graphical overview of the test case results is created (usually the test result is printed to the console). In this case, the class properly handled the null pointers and threw the expected NULL_EXCEPTION exception. The graphical overview from Eclipse is shown in Figure 6.

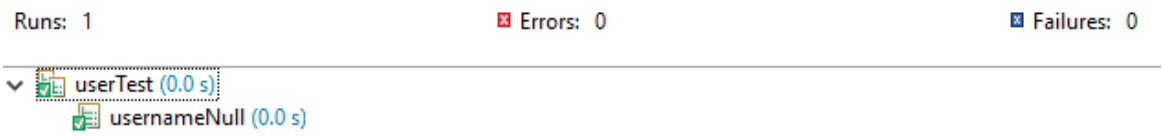

Figure 6: Example gtest Results
Description: The results displayed in the "C/C++ unit" output in Eclipse show, that no errors or failures were found during the gtest test run.

Firmware can be expanded with integrated (white-box) tests to supply white-box test results via JTAG or other debug interfaces, even though technically JTAG is designed to be used for black-box tests. An alternative approach is to test the firmware within an emulator, where results can be collected via (virtual) IO ports [45].

Hardware white-box tests are less developed. However, where hardware description languages (HDLs) are used, at least theoretically, even though hardly practiced, white-box testing through formalized tests is possible [46].

Suppliers often shy away from independent white-box tests, because they would need to deliver the full system design and implementation to an independent party, which increases the risks associated with industrial espionage. Therefore, white-box tests are mainly conducted by internal test teams or in isolated rooms at the supplier's site.

White-box tests can be highly effective in testing of input validation and logic correctness, as the functions can be directly parametrized and tested against a big set of input-output-combinations. However, if the testers are too dependent on the program structure defined by the programmers,

they might not find defects that are result of the structure of the program instead of an individual unit. In conclusion, white-box tests can help to identify faults that are currently not necessarily propagating to the outside, but that puts the system in an undesirable state (for example an infinite loop that is only triggered for certain inputs).

Road Vehicle Specifics

Because of the often-complex supplier contracting, source code is not always available and therefore white-box tests are not always possible. However, it can be useful to use white-box testing to identify situational failures, that only occur, when very specific, hard to recreate conditions are met, for example, if an emergency call is made.

4.3.3 Analytical Tests

Analytical tests (also called static analysis) are tests, where the system is not in operation, but instead is tested for a set of static properties. These properties could be use of descriptive loops instead of goto statements, restrictions in use of deprecated functions, use of interior traces on circuit boards for critical information or policies for usage of third-party components.

A set of static system policies is often specified under the belief that certain practices, methods, technologies, etc. are so problematic, that they are expected to result in components with inacceptable security. Automation can often be used to verify that the policy is fulfilled for all components. For example, a search for usage of the unsafe gets() function could be implemented directly in the SDK.

Most software development processes are already supported by a set of analytical tests from the compiler. A compiler warning is thrown if any problems are found. However, for some projects additional quality assurance policies exist, so custom automated or even manual analytical tests suites might be conducted.

For hardware, it is much harder to test for static properties if no HDL is used designing it. Nevertheless, some properties, like usage of external hardware components can be monitored (even though usually manually).

Analytical tests are usually result of past negative supplier experiences with certain system designs and are therefore often supplier-specific. Analytical testing can help to ensure that expectations for code quality and hardware design are met.

Road Vehicle Specifics

In road vehicle software development, often low-level programming languages are used to handle the high amount of IO devices. These programming languages are prone to unsafe code constructs and therefore benefit more from analytical tests.

4.4 System Security Assessment

The system security assessment validates the security functionality in terms of its practical ability to mitigate expected risks. In comparison to the security design inspection and functional security test, it is much more independent from the convictions of the designers and developers, but achieves a lower fault detection quota (coverage). System security assessment cases should be identified by a security assessment analyst and peer-reviewed. To verify that the project security efforts caught all relevant vulnerabilities, the assessment cases should answer at least the following questions:

- Does the security functionality protect against all common attacks?
- Are all non-valid inputs correctly rejected or dropped?
- Is critical information transmitted over external or accessible connections secured?
- Is there a secure fallback if depending functions are impaired?

- Does privilege and access management properly restrict user access?
- Are additional services available, that are not considered in the specification?

In the following, a set of typical assessment techniques, with no claim of completeness, will be described.

4.4.1 Known Exploit Tests

To identify if a system is vulnerable to common attacks it is feasible to test the system against an exploit library, as developers repeatedly make the same mistakes and testing against an exploit library finds these vulnerabilities [47](p. 84). This approach is also known as vulnerability scanning [48](p. 4-4). There are many tools for vulnerability scanning available on the market, therefore a threat library does not have to be maintained individually. Any risk that gets identified through a market-available vulnerability scanner, should be considered to be remediated, because it is likely that an attacker finds and uses an exploit that is discoverable through automated tools.

Road Vehicle Specifics

Few exploit collections for road vehicle information systems exist outside of the hacker inventories. However, most of road vehicle information systems rely on third party components, like operating systems and drivers, for which such libraries might exist.

4.4.2 Invalid Input Tests

Insufficient input validation can lead to component or system crashes and consequently have a high impact on the availability of the system. Invalid inputs that are carelessly processed can cause undefined system behavior including buffer overflows as well. In these scenarios, a loss of confidentiality, integrity and availability occurs. Figure 7 illustrates how input validation protects the program logic from malformed or even malicious inputs and therefore reduces the risk of unintended system behavior (outputs).

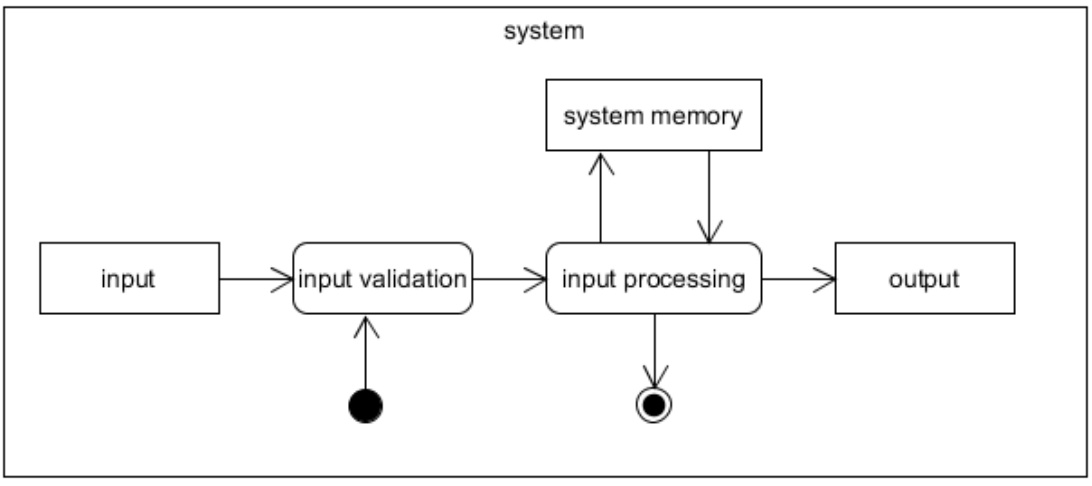

Figure 7: Input Validation
Description: The figure shows how input must first pass an input validation layer before it is processed. The input processing might also use additional information from the system memory and then produces an output and/or stores information in the system memory. Input that is dropped during input validation will not affect system memory or output.

The invalid input tests described in this chapter are the elevation of the functional testing techniques described in chapter 4.3 to the interface level. Instead of a single system component, an external interface is supplied with inputs of mixed validity. The system is then observed for unusual behavior (output), most notably system or interface crashes and unresponsiveness.

Mixed validity inputs can be generated with fuzzers. Tests in which a bulk of those inputs is supplied to the system are often referred to as “fuzzing tests”. Fuzzing tests can and should be executed for external interfaces as well as user-manipulatable files to verify the sufficiency and correctness of input validation.

Some primitive fuzzing implementations supply pseudo-random data frames to an interface. These implementations are not effective, if authentication is required to transmit data via an interface or if elaborate handshakes are used to establish a connection. Most input will then be (correctly) dropped in the first layer of input validation (which is often the validation of credentials). Therefore, the most successful fuzzers create fuzzed inputs based on “complete knowledge of the layout and contents of the inputs” [49]. Some commercial-off-the-shelf fuzzers solve this problem through data and state modelling or act as a man-in-the-middle in a valid connection [49] [50].

Unfortunately, it is often required to fit the fuzzer for the system under assessment, although fuzzers are available for most customer-relevant interfaces and many industry standard interfaces including the ones of the road vehicle industry [51]. A security assessment analyst or even a dedicated developer would need to make these adjustments to the tool.

Road Vehicle Specifics

Malicious input is expected on all non-vehicle-internal interfaces of road vehicle information systems. This applies especially to Bluetooth, USB, WLAN, OBD II and mobile data. Invalid input tests should be conducted at least for these interfaces.

4.4.3 Service Identification Assessments

An additional available service (externally accessible component) in the system does not necessarily contain a vulnerability. Nevertheless, such a service is problematic, as the service has never been specified. It might offer additional interfaces, information storage or program logic whose risks have not been analyzed. If such services are identified, it is likely due to a lack of specification or because third party components offer more services than used by the system.

As a proper risk analysis has not been conducted for these services, the risks are unknown (and might be significant). Especially if the service comes from a third-party component, for which no update process is in place, it is possible that the system will be vulnerable to attacks against future vulnerabilities in these third-party components.

Luckily many automated proprietary and open-source tools for service identification are available for common interfaces, especially the IP interface and common operating systems. For less common interfaces and operating systems (or any other system in which “services” are of interest like containers, virtual machines, etc.) a manual assessment might be required.

Road Vehicle Specifics

Because of the limited reach of road vehicle networks and the low user configurability of the systems, unspecified open ports with reachable services are untypical. Thus, it is unlikely this assessment will find faults in a road vehicle information systems.

4.4.4 Data Transfer Assessments

Most systems transfer data to other systems or external storage. These connections can potentially be eavesdropped and, if no protection is in place, the eavesdropper can collect the data the system transferred in plain text. In some cases, the risk associated with eavesdropping of the transferred data is so low, that securing the connection is not cost-effective. This especially applies, if the connection medium is physically limited, so an attacker would need physical access to the systems location such as USB connections. Transfer of data with differing security needs can result in mixed traffic (protected and unprotected) on the same connection.

Data transfer assessments can be useful to identify, if the data transmitted has obvious protection issues. As there can be data with different confidentiality, integrity and availability needs transferred via the same connection, the decision if such an obvious issue is present can be challenging and is ultimately up to the assessor and a proper risk analysis. The decision is much easier, if a general policy of the data transfer protection is in place, for example a project-wide policy that HTTP is forbidden and only HTTPS shall be used.

Only in cases where such policies exist, an automated test can be performed. Otherwise, it is up to the assessor to identify inadequately secured data transfer. In both cases, it is required to sniff the outbound connections of the system and capture data traffic for later analysis. Such sniffers are available for most (non-proprietary) connection protocols.

Road Vehicle Specifics

Positional data, as well as driving habits might be personally identifiable information and should be handled with caution. If such data can be eavesdropped on external interfaces, immediate action should be taken. Other security-relevant data might be transferred, especially within vehicle manufacturer backend communication.

4.4.5 Tamper-Resistance Assessment

In some cases, the system is property of the customer, while distributor information is stored in the system that should not be manipulated. This includes information such as public update keys, unlock codes for additional system features, assigned numbers (e.g. the MAC address). A customer might have an interest in tampering this data against the interest of the distributor (e.g. by unlocking an additional functionality without purchasing it). This can be achieved by a security functionality that validates the origin of the information, before the program logic processes it. This is done by encrypting the information itself or a hash of it (creating a certificate) with a private key on the distributor's side. The program logic then decrypts the information or verifies the hash/certificate on the system. This process is depicted in Figure 8.

Distributors often store the public key(s) in dedicated hardware together with the algorithms to encrypt and decrypt such data. The most common implementation is the Trusted Platform Module (TPM) standard [52] but other hardware security module (HSM) architectures exist. Because of the logical and physical separation of the key-handling from the system and the certified security of the HSM (usually FIPS-140 or Common Criteria), this public key is assumed to be valid and serves as a root of trust. Based on this design, many different security functions are derived including Secure Boot (boot chain of trust) implementations. Implementations that rely on the HSM can still be vulnerable and often distributors want to verify if the system can be tampered in a way that would allow to bypass these validity checks.

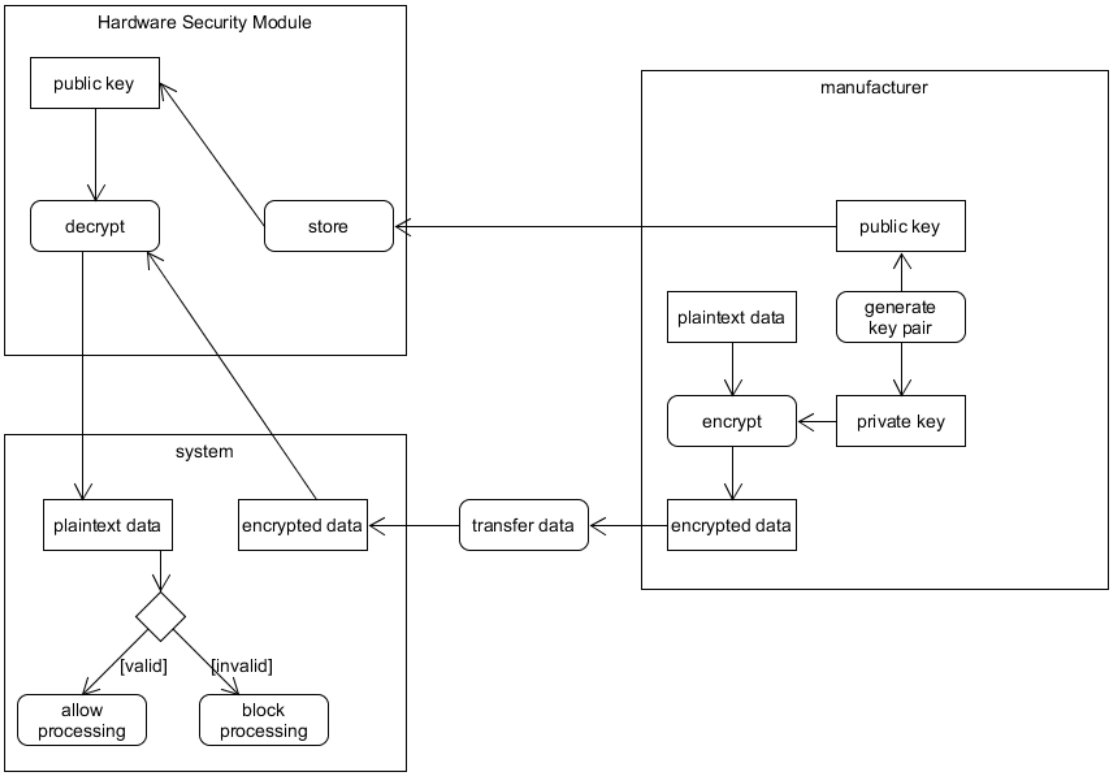

Figure 8: HSM Validity Check
Description: The manufacturer encrypts the data with his public key and stores it on the system. He supplies the public key to the HSM, which decrypts the encrypted data on system request. If the plaintext data received from the HSM is still valid (i.e. still makes sense within the program logic), the data is processed. Possible attack vectors would be changing the temporary plaintext data instead of the permanent encrypted data or exchanging the program logic that checks the validity of supplied data (if only a certificate is checked) with one that defaults to "valid".

In this case a white-box test of all HSM-related components (those who call the API of the HSM) would be part of the functional security test. Additional penetration, fuzzy, or other tests might be useful, as the logic behind validity and tamper detection functions is often quite complex and therefore prone to error. The decision, to which extent additional tests are performed, should be based on the security design of the system, which could even reveal possibilities to make the system more tamper-resistant without extensive validity checks on the system. For example, instead of storing additional (deactivated) features on the system, they could be downloaded, if needed. As the customer does not have physical access to the server, fewer security measures would be required. Subsequently secure validation of the activation code on the distributor's server would be significantly cheaper to implement than validating it locally.

Road Vehicle Specifics

Activation of premium (infotainment) packages is a popular business model within the automotive industry. Especially within this context, tamper-resistance assessments might be useful.

4.4.6 Dependency Assessment

Proper behavior of security functions is often depended on secondary components. For example, an authentication mechanism requires a valid credential data set to verify received credentials. If the dependencies are not met, a security function should switch to a secure fallback state. This state is usually stricter than the operational state and should at least ensure the confidentiality and integrity of stored information. For the authentication example, this would mean, that no credentials

are verified until the credential data set is available again and that no credentials are insecurely stored. The number of dependencies between components grows quadratically with the number of components, as any component is potentially able to affect any other. Subsequently, a specification of the fallback behavior often doesn't exist.

As not every fallback behavior is specified, not every fallback behavior is tested during the functional security test. During the system security assessment, individual components can be intentionally disabled to assess the fallback behavior of the security functionality. This can reveal additional dependency-related security faults. A representative set of functional security test cases from different security functions can be chosen and repeated with a deliberately disabled component to verify the correct behavior of the functions, while some dependencies are not met. The test effort would then scale linearly, as long as the amount of representatively chosen test cases is constant.

Road Vehicle Specifics

ECUs have additional requirements, when it comes to fallback mechanisms. When a component fails, dependent components need to provide basic functionality until the vehicle has stopped. Therefore, the fallback is often relatively tolerant to malformed inputs. Temporary vulnerabilities might arise from this behavior.

4.4.7 File Permission Assessment

Operating systems based on Windows NT, UNIX or Linux feature a file access control system based on roles and groups [53] [54]. As access permissions of files can be individually chosen by developers and administrators in these operating systems, vulnerabilities related to manipulability (cf. chapter 4.4.5) or information disclosure are common. Subsequently, tools have been introduced to find files with problematic permission settings.

The most prominent tools are "AccessChk" [55] and "AccessEnum" [56] for Windows as well as the "namei" tool, which is included in most Linux distributions through the util-linux package. With proper parametrization, a security assessment analyst can identify common file access privilege faults in the system through such tools. The assessor should judge the permissions based on the least privilege principle, meaning that "every program and every user of the system should operate using the least set of privileges necessary to complete the job" [57]. Through the tree hierarchy of files and folders, file permissions are often inherited from higher-level folders. This enables relatively fast assessment of file permissions in structured systems. Assessing systems with a fragmented file structure is significantly more difficult.

Road Vehicle Specifics

In road vehicles, multiple types of information systems are present. Infotainment systems are usually using Linux, while other ECUs often use real-time operating systems. Real-time operating systems often don't feature a file access control system. Thus, file permission assessments might be limited to the infotainment systems.

5 ASSESSMENT EXECUTION

The assessment plan shall outline the execution process. Some execution aspects that are especially important will be more specifically described in this chapter. Results must be documented to gather a data basis for reporting. The documentation process is described in chapter 5.1. Some project progress indicators should be collected during the assessment, which is described in chapter 5.2. Even a well-planned assessment might need to be adjusted, if slow progress or poor detection rates are identified from these indicators or stakeholder feedback. Chapter 5.3 describes, how these adjustments can be made, while not compromising the overall assessment goal.

5.1 Assessment Documentation

Assessment cases should be within 3 different states: “passed”, indicating the security functionality lived up to the expectations, “failed”, indicating there is a fault within the security functionality, or “open”, indicating that the assessment case has not been conducted (yet). These states might fragment into additional sub-states, to assign more specific workflows to them. For example, “blocked” could be a sub-state of “open” to indicate that a precondition cannot be established yet.

When an assessment case has been conducted, the result must be documented. For passed assessment cases, it is only required to note that they did so, as the assessment run is already understandable by the preconditions, actions and expected behavior. For failed assessment cases, a description of the fault must be documented, so the deviation is adequately described.

On the software implementation level, the fault description is, in the broadest sense, a bug report. It should enable the developer to fix or, if a fix cannot be delivered, at least document the fault, so it can be avoided or addressed on a business level.

The information pieces most often used to fix a bug are “steps to reproduce, observed and expected behavior, stack traces [logs], and test [assessment] cases” [58]. At least this information should be given to the supplier. In agreement with the supplier, additional information pieces specific to the project might have to be delivered. For example, screenshots can be useful for reporting user interface faults. On a design or analytical test level, a detailed description of the current solution, the issues with it and an alternative solution should be provided as a fault report.

If the documentation needs are higher than what’s described in this paper, ISO standard 29919-3 “Software and systems engineering -- Software testing -- Part 3: Test documentation” [59] might help to introduce a structured approach to documenting all assessment activities.

All results should be delivered to the supplier as soon as possible, or even maintained directly within the supplier’s bug tracking / ticketing tool and/or knowledge base (cf. chapter 3.5.1).

5.2 Resource Tracking

All expenses should be tracked by the security assessment manager. This is usually done by assessment team members reporting their working hours and material consumption into a tracking tool. The security assessment manager is in charge to provide adequate reporting avenues for all team members and occasionally review the conclusiveness of reported data. Together with the assessment documentation, this enables the security assessment manager to comprehend inefficiencies in spending, as well as planned-to-actual expense correlations. Qualitative and quantitative data on assessment progress and resource expanses enable effective steering of the project.

However, overreactions, because of limited deviations, should be avoided, as impulsive reactions might hurt the project more than the initial deviation. Where action is inevitable, the following chapter gives guidelines on assessment adjustments, that help to improve the assessment progress, effectiveness and efficiency.

5.3 Assessment Adjustments

In chapter 4.1 the definition of a successful security assessment has been introduced as one that can (within a reasonable budget), identify security faults as early and completely as possible.

However, often some assessment cases don’t yield the success expected, meaning that test cases do not identify enough security faults, do not identify them early enough or only do so with unreasonable effort. There currently is no verified findings on what the main reasons are, that assessment cases need to be reworked or scrapped. However, as the concept of software testing assumes, that human error will always be present in complex endeavors, it is consequent to assume it will be in the assessment planning and execution as well.

If a set of assessment cases has been identified as not applicable or otherwise inappropriate, any adjustment must still maintain the guarantee, that sufficiency and correctness of the security functionality is adequately assessed. Usually, this can be done, by replacing the set of test cases by some that yield the same level of assertions, meaning that the same failures and therefore equivalence classes are covered by the new test cases.

If adjustments shall be made to reduce costs, it must always be controlled if the cost reduction justifies the expected effectiveness reduction. In such cases, the decision to drop such test cases or to replace them by more cost-effective ones, doesn't contradict the security functionality root requirement of sufficiency and correctness introduced in chapter 2.5. All assessment adjustments should be properly documented, especially through versioning of assessment cases and reported to all stakeholders.

An additional challenge is the handling of change requests. This is quite a complex topic, which depends on the current project state, development lifecycle, quality of change request, management willingness to react to them and dependence of system components. Therefore, the general directive, that sufficiency and correctness of the system's security functionality needs to be reassessed with every change, must suffice at this point. Nonetheless, publications on the topic are available [60] [61].

6 ARTEFACTS AND REPORTING

Assessment results are used in multiple ways. During development, the findings are used to supply the developers with information about implementation and design choices needing improvement. After launch, the system assessment results can serve as proof for a high quality, secure system to potential customers. Therefore, a summarizing report or even a certificate should be created for marketing, legal or management purposes. If major deficiencies are mentioned in this report, the distributor might also refuse to accept the system delivery from the supplier. As positive, as well as negative, results are quite impactful, creation of credible and clear reports is an important task for any assessment. Within this chapter, recommendations about reporting and achieving certifications is given.

6.1 INTERMEDIATE RESULTS

Failed assessment cases should be reported immediately to the supplier. They should be documented as failed until the supplier delivers a fix for the fault and a proper reassessment has been conducted. This applies to all 3 assessment stages. The security design inspection, the functional security test and the system security assessment. Early reporting of intermediate results enables the developers to fix the fault, while the complexity of the system is still low, which ultimately reduces development costs. This is usually done via a routinely created extract from the assessment tracking or could be integrated into the supplier's fault tracking tool. The second option is appealing, as the developers can instantly start fixing found faults, but the results must be tamper-resistant if the security assessment summary is created using the same data.

Results of assessment cases should be noted immediately after execution for later reporting. Proper proof for the result in the form of logs, traces and files should be stored as well (cf. chapter 5.1). This helps especially in cases, where liability claims are made towards the supplier or test provider.

6.2 SECURITY ASSESSMENT SUMMARY

The security assessment summary should inform if the system has security functionality that is capable to sufficiently and correctly protect the system's components and information stored in it. It is combined from the results of the security design inspection, the functional security test and the system security assessment.

It should present with which methods the security design artefact has been created and if the system has been adequately modeled in terms of the risk associated with the system. Where flaws in the

model exist, an estimation of coverage and/or the unmitigated risk associated with it should be given. A list of faults with the system security design should be appended to this overall result.

For the functional security test, a quantitative overview of the results should be given and rated by the missed risk level reduction identified in the security design artefact. A list of the executed test cases and their results should be appended to this overall result.

For the system security assessment, an overview of conducted assessment types and their result should be given. Found, unfixed faults can be reported by their amount and can be complemented by an estimate of their risk level. The limits of the assessments executed should be described, especially by the faults that could not have been found with these assessments. A list of all faults found during the system security assessment, that have not been fixed by the supplier should be appended to this overall result.

As a closing result a textual description or calculated value for the difference between the functionality that could have been cost-effectively implemented (with a description of what is assumed to be cost effective) and the implemented functionality should be given.

6.3 STANDARDS AND CERTIFICATION

When the extent of useful security functionality has been identified through risk analysis, it might be useful to check, if existing security standards are covered by the security functionality identified as cost-effective. If such standards exist, the distributor might be well-advised to aim for certification(s) based on such standards to prove to a (potential) customer that the implemented security functionality is state of the art for the system type. Some minor adjustments might need to be made to the security design to comply with all requirements of the standard. In general, standards that require much more security functionality than identified as adequate during risk analysis should be discarded, as the additional expenses do not lead to a significantly more secure system. This only holds true, if there is no exceptional market demand for accordingly certified systems.

For example, products can be certified per the Common Criteria standard for a certain protection profile. A protection profile is a set of requirements towards a category of products. Many more, mainly niche standards exist and regularly come with a certification process. Often evaluators need to be licensed by a national or international authority to provide such certification (the national IT security agency in the case of Common Criteria [62]).

If certification is desired, the certification procedure should be incorporated into the development and testing plans. Usually certification can only be achieved for the final product and systems therefore are often released before the certification has been issued. As an alternative, the release of the product would need to be delayed, which is often associated with high opportunity costs, as revenue is only generated after release.

In general, two steps need to be taken to include an additional standard. First, the security design needs to incorporate the security functions required by the standard. Second, the test strategy needs to plan the assessment activities required by the standard, possibly leading to a certification. However, depending on the standard in question, the actual adjustments might be more complex and the actual implementation steps need to be elaborated.

Road Vehicle Specifics

For many safety-critical road vehicle information systems ISO 26262 “Road vehicles -- Functional safety” [62] is applicable and even though no specific certification process is in place, compliance with different Automotive Safety Integrity Levels (ASILs) can be claimed within two levels of independent testing. However, the standard is rather broad and requires the distributor to evaluate not only security-related failures, but all other system failures as well. Therefore, a bundling of different assessment domain results is required.

7 COMPARISON TO OTHER APPROACHES

The approach presented in this paper does seek to maximize stakeholder value of the product by cost-effectively reducing risk. However, other approaches to security exist of which multiple will be introduced in this chapter. All these approaches have advantages and disadvantages, the main one of the risk-driven approach being, on the plus side, the maximization of net profit. On the downside, non-tech-savvy customers, might not be able to identify the security benefit of the system, which is something that governments should and mostly do compensate by liability laws [63] [24]. Some of the other approaches will be described in this chapter and advantages as well as disadvantages of them will be identified.

7.1 STANDARD-DRIVEN SECURITY ASSESSMENTS

There are many applicable standards for information security and the testing of it. Most notably this is the Common Criteria standard [23], but other (industry-specific or system-class-specific) standards might also be applicable. Where compliance to a standard is required by the customer or by local regulations, these standards should obviously be included in the system specification (or system security design).

Often the certification of the system, based on a recognized standard, is the only driving factor for the security of the system. In these cases, the distributor would not conduct a proper risk analysis but instead only applies the standard for the desired certification. The main problem with this approach is, that it remains unclear if the security functionality does match the security needs of the system.

This can lead to an implementation as depicted in Figure 9. Security functionality has been implemented that is not cost-effective and some has not been implemented that would have been cost-effective. Nonetheless, the distributor could instruct an independent or internal, depending on the standard, assessment of the functionality and would achieve a certification of the system.

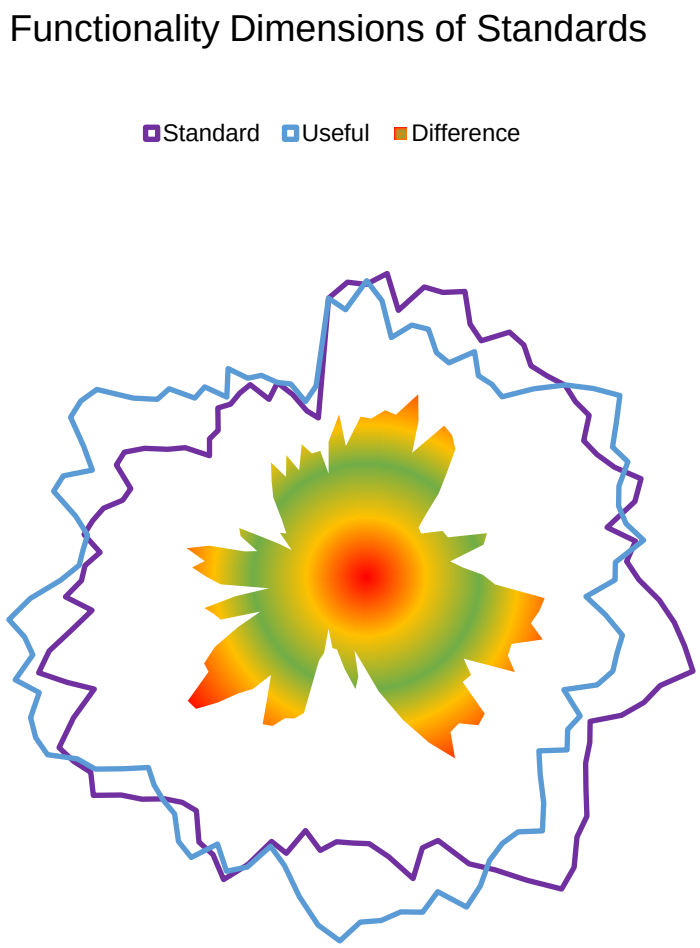

Figure 9: Functionality Dimensions of Standards
Description: The blue line represents the implementation cost (as the distance from the center) into a security function to mitigate a risk (as the angle from the South) suggested by risk analysis, while the purple line does the same for a standard. This representation is called a spider graph. The plane in the middle is the difference between the implementation costs of the standard-suggested and risk-analysis-suggested function. Ideally the standard-suggested functionality would always match the risk-analysis-suggested one. The plane would then be completely green on the edges. However, often a too expensive or too cheap function is suggested by the standard (both represented by yellow to red edges of the plane).

There are 3 main goals when applying standards: Cost reduction, introduction of an adequate security level, and competitive advantages [64]. Out of these, the first two can be more efficiently achieved with proper risk analysis and an adequate security functionality based on it, as shown before. So, the main advantage of applying standards is the competitive edge certified security provides. With certified security, the distributor and supplier can prove proper security in case of lawsuits, use the certification for marketing purposes or compete in competitive tendering procedures where such a certification is required.

However, the benefits of both, risk-driven and standard-driven security, can be combined by first identifying the risks of the system and what security functionality would be cost-effective and then identifying if there are recognized standards for which a certification could be achieved (cf. chapter 6.3). The certification should require little additional effort, but should have marketing, legal, and commissioning advantages.

Nonetheless, if the customer has no means of comparing the actual security level of market-available systems against each other and liability for security breaches is not shifted to the distributor, the distributor would act economically reasonable if he conducts only standards-driven

security practices. In the market overall, this would still lead to an economic inefficiency, which is a condition that should be lifted by new regulations or industry agreements, including new means of certification.

7.2 Recognition-driven Security Assessments

Another approach is the exclusive usage of exploit libraries and service identification tools to assess the security of the system. These are black-box tests, where the presence of common system vulnerabilities is verified or falsified. The main advantage of this approach is that most high-probability risks are recognized and can be mitigated, while the overall test cost is very low, because of the possible automation. The downside is, that risk that are exclusive to the system under test or very uncommon may go unnoticed.

Therefore recognition-driven security assessments are most beneficial for systems in maintenance status, not development status, as a cyclic automated test (possibly over the whole IT infrastructure), where exploit and service libraries can be frequently updated and limited documentation and source code is available. Maintenance however, is out of scope of this paper and because of the obvious limitations to identify system-specific risks, the recognition-driven approach can be disregarded as too limited for systems in development.

7.3 Customer-Request-driven Security Assessments

Some distributors only implement security functionality as requested by the customer, mostly to keep development costs low. Exclusively functional tests are then conducted to verify the correct implementation of the customer requirements. The customer must be assumed to have little knowledge about the risks associated with use of the system and therefore it is likely that the security functionality requested by the customer is inadequate for the system to be developed.

It is therefore up to the distributor to suggest adequate security functionality to the customer after an initial risk analysis and he should inform the customer about the drawbacks associated with not conducting proper risk analysis, namely: High probability for security incidents related to the system, development costs for useless security functionality and potential legal risks associated with insecure system development.

If the customer accepts these expected negative consequences and still wants to have his requested security functionality implemented, that must be accepted. However, no dedicated security test would then be needed, but the regular functional test should suffice, as sufficiency of the security functionality will not be achieved.

7.4 Penetration Test

Penetration testing is the “process of attempting to gain access to resources without knowledge of usernames, passwords and other normal means of access”. However, “it’s important to understand that it is very unlikely that a pen-tester will find all the security issues” [65].

Therefore, penetration tests are an additional measure to verify that for systems with already good security posture, an attacker is unlikely able to develop an exploit. It is possible to integrate a penetration test in the system security test described in this paper, but penetration tests should always be the last test to identify problems within all previous implementation activities and never be conducted as the only test to verify the system’s security.

It should be noted that penetration tests are, in the terminology of this paper, an assessment, not a test, because no plan on how to find vulnerabilities is pre-defined (cf. chapter 2.1). However, as the term penetration test is used in general parlance, it has been used in this paper to simplify comparability.

8 CONCLUSION AND OUTLOOK

The risk-driven approach to security assessments focuses on cost-effective security functionality instead of abstract security targets. It matches a “Security by Design” development lifecycle with an assessment that verifies the sufficiency and correctness of the security functionality implemented. The assessment is organized in three stages. The security design inspection assesses, if the security design artefact describes a sufficient security functionality for the system. The functional security test assesses, if the security functionality has been implemented in accordance to the security design. Finally, the system security assessment is conducted to verify, that the implementation of the functionality in fact increased the security level of the system.

With the risk-driven security assessment approach, it is therefore possible to give the supplier the information on what the security-critical faults of the system are. Proper documentation enables him to fix them. A company or team following the risk-driven security assessment approach can therefore offer attractive services to the stakeholders involved in the system development process.

8.1 SUMMARY OF ASSESSMENT TECHNIQUES

Within the context of this paper, three assessment stages (cf. Figure 10) have been defined. For these assessment stages, different assessment techniques can be applied. The assessment techniques described within this paper do not claim completeness, but all assessment stages should have appropriate assessment cases associated with them.

Figure 10: Test Stages
Description: The risk-driven assessment of security functionality consists of three test stages. The security design inspection judges the sufficiency of the functionality described in the design artefact. The functional security test judges if the security functionality has been implemented correctly and the system security assessment verifies that the assessed sufficiency and correctness of the system translates into a reduced risk level and reveals improvement possibilities.

The assessment techniques should be chosen based on what is expected to fit the system under assessment. The assessment techniques described in this paper as well as their benefits and downsides are summarized in Table 9 below.

Assessment Technique	*Benefits & Downsides*
Security Design Inspection	
Criteria Catalogue	Pro: The actual inspection requires limited skill. Con: Some contextual design faults might not be detected.
Moderated Discussion	Pro: High expertise assessment of the security design; Designers improve their skillset. Con: High workload for security designers.
Functional Security Test	
Black-Box Test	Pro: Faults that result in a failure are found for simple inputs. Con: Masked faults or faults that only result in a failure with specific input combinations are not found.
White-Box Test	Pro: Masked faults can be found; Inputs can be supplied at a high rate, therefore specific input combinations can be tested. Con: Low independence of the tester might lead to biased results. High effort for limited customer value.
Analytical Test	Pro: Helps to enforce supplier and distributor policies and to improve overall implementation quality; Highly automatable. Con: The implementation quality might not offer any customer value.
System Security Test	
Known Exploit Tests	Pro: Most common attacks will be mitigated; Low effort. Con: Does not account for targeted or completely new attacks.
Invalid Input Tests	Pro: High effectiveness in identifying risks associated with replay and race condition attacks; Low effort. Con: Limited applicability for multi-layered input validation systems.
Service Identification Assessment	Pro: Can identify additional (unspecified) components, that would need reassessment. Con: Stays superficial in complex systems.
Data Transfer Assessment	Pro: Covers the attacker's main information supply if sufficient traffic is produced. Con: High amounts of manual work; Depends on the availability of communication endpoints.
Tamper-Resistance Assessment	Pro: The capabilities of an attacker with physical access to the system can be assessed. Con: Usually tamper-resistance is not necessary, because information could be stored elsewhere.
Dependency Assessment	Pro: Dependency-related faults, that are not identified otherwise, can be found. Con: High effort; Dependency-related vulnerabilities are only temporarily exploitable.
File Permission Assessment	Pro: Cost-effective assessment to verify the compliance with the least privilege principle in structured systems. Con: Not feasible for fragmented systems.

Table 9: Summary of Assessment Techniques
Description: The table summarizes the assessment techniques described in chapters 4.2 through 4.4. The benefits and downsides are briefly named.

While the choice and description of assessment cases should be done by a security assessment analyst, the actual execution can often be performed by a tester to reduce labor costs.

8.2 PENDING CONSIDERATIONS

There are some underlying limitations in the methods used to retrieve the information in this paper. No primary sources (measurements, interviews, etc.) were used within this paper, instead it relies on secondary sources (literature; cf. chapter 1.2). This allowed to give a general overview of the

whole security assessment process within a reasonable timeframe, but all conclusions are limited to the state-of-the-art and could not show mistakes within current convictions.

It is therefore important to prove or disprove the recommendations given, based on quantitative data collected by business data analysis or scientific research. For the following topics, no or only unreliable quantitative data was available at the time of research:

- Customer willingness to pay a premium for secure products
- Successful liability claim quotes for security faults
- Fault detection likelihood of assessment techniques
- Average impacts of security attacks by attack type

These are urgent research areas, not only for verification of this paper, but for the security assessment business overall, as these topics significantly affect risk estimates.

At some points within this paper, specific improvements possibilities have already been identified. These improvements are usually adoptions of more specific approaches (standards). As the methods outlined in this paper often suffice for smaller projects, these improvements can be made after the initial adoption of the risk-driven security assessment approach.

More security assessment techniques can be integrated into the outlined assessment stages. The exploration and adoption of additional assessment techniques can help to improve fault detection rates and to reduce assessment costs. Overall, the findings enable the introduction of a minimalistic, later extensible, information system security assessment process or services based on it.

9 APPENDIX

9.1 VOCABULARY

In security testing, as in IT in general, terms and definitions are often used different than in non-IT contexts or the same term may refer to multiple definitions or concepts. Consequentially, it is required to have a clear definition of terms used in this document to keep declarations consistent and the meaning of statements clear.

This paper tries to be consistent with the vocabulary used in ISO/IEC 15408-1:2009(E), which is referred to as Common Criteria [23]. In some cases, consistency with this standard is not preserved, because other terms are significantly more descriptive. This especially applies in cases where this paper refers to test activities, while the ISO standard refers to evaluations (i.e. assessments; see chapter 2.1). As an additional source of definitions, a general and economic dictionary has been used [8] [66].

Where gender-specific terms are used for readability, the statement applies to all genders (and gender identities). Where specific products, services or companies are mentioned, the statement applies to all competitors and their comparable products and services.

In the following, important terms and their definitions used in this paper are listed.

Term	*Definition*
Artefact	A product resulting from a process.
Assessment	The process of determining if specific objectives are met; A judgement if specific objectives are met.
Attack	A malicious attempt to exploit a vulnerability of the system.
Authenticated	Doing something, while previously having delivered proof of identity or trustworthiness.
Authorized	Doing something in a relation of previously established trust.
Bias	Preference of a belief for subjective reasons.
Boundary value	The highest or lowest value of an ordered set of values.
Bus	A data transfer system, usually described by the medium and the protocols.
Component	A cohesive part of the system only communicating through interfaces.
Design	A description of the way something is planned.
Entry criteria	The state that must be achieved before something can start.
Exit criteria	The state that must be achieved before something can stop.
Exploit	Code or data that takes advantage of a vulnerability to cause unintended system behavior.
Fault	A flaw in a component or system that can cause it to fail to perform its required function [30].
Failure	Deviation of the component or system from its expected delivery, service, or result [30].
Function	A capability of the system.
Functionality	All functions of the system.
Information system	Software and/or hardware transforming input signals to output signals and eventually storing information.
Interface	A termination point of a component or system that is intended for data exchange.
Logger	A device recording system event reports (logs).
Off-the-shelf	Something that is instantly available in the market.
Penetration	Forcing access to components that are protected by security functionality.

Protocol	A formal agreement about the communication structure between connected components and systems.
Resource	A medium that is used in the production of goods or services and that is assumed to be limited in supply or associated with a cost.
Risk	An event, where expected and actual benefit differ.
Risk level	The product of the impact and likelihood of a future event that creates a difference between the actual and expected outcome of an activity.
Safety	The reliability of a system's ability to remain in a state, in which it does not harm its environment.
Security	Protection of a system against external incidents that are assumed to have a negative impact on the system itself.
Security fault	System imperfection that negatively impacts the security of the system, if specific conditions are met.
Security incident	Occurrence, where a stakeholder has a negative impact, because of a security failure.
Signal	An informative transmission.
Sniffing	Observation of data transmissions.
Specification	A detailed description of how something should be done.
Stakeholder	A person or organization with interest or responsibility for the success of the system development endeavor.
Strategy	A detailed plan for achieving success.
System behavior	The system's matching of inputs and outputs.
Test	An effort of discovering, validating or quantifying the suitability, properties or performance of an object through a pre-defined plan.
Test case	An activity within a test to validate or quantify a single property, performance metric or suitability for a single application of an object.
Timeboxing	Limiting the amount of work to be done for a subject.
Validation	The process of proving that something fulfills a demand.
Verification	The process of proving that something occurred as intended.
Vulnerability	An exploitable security-related system fault.

9.2 ABBREVIATIONS

Abbreviation	*Meaning*
ABS	Anti-lock braking system
API	Application Programming Interface
DHCP	Dynamic Host Configuration Protocol
DoS	Denial of Service
ECU	Electronic control unit
ESP	Electronic stability program
EUR	Euro
FIPS	Federal Information Processing Standards
HDL	Hardware description language
HMI	Human Machine Interface
HSM	Hardware security module
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IEC	International Electrotechnical Commission
IO	Input / Output
IP	Internet Protocol
IQSTB	International Software Testing Qualifications Board
ISO	International Organization for Standardization
IT	Information technology
JTAG	Joint Test Action Group
LOC	Lines of code
LIN	Local Interconnect Network
MAC	Medium access control
MOST	Media Oriented Systems Transport
OBD	On-board diagnostics
OSI	Open Systems Interconnection
PC	Personal computer
RFC	Request for Comments
ROI	Return on investment
SCADA	Supervisory, control and data acquisition
SDK	Software Development Kit
SQL	Structured Query Language
TPM	Trusted platform module
UML	Unified Modeling Language
USA	United States of America
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
XP	Extreme programming

9.3 LITERATURE

[1]	J. Stamp, J. Dillinger, W. Young und J. DePoy, „COMMON VULNERABILITIES IN CRITICAL INFRASTRUCTURE CONTROL SYSTEMS," Sandia Corporation, 11 11 2003. [Online]. Available: http://energy.sandia.gov/wp-content/gallery/uploads/031172C.pdf. [Zugriff am 22 01 2018].
[2]	B. W. Boehm, „Guidelines for verifying and validating Software Requirements and Design Specifications," 1979. [Online]. Available: GUIDELINES FOR VERIFYING AND VALIDATING SOFTWARE REQUIREMENTS AND DESIGN SPECIFICATIONS.
[3]	G. J. Myers, The Art of Software Testing, New Jersey: John Wiley & Sons, Inc., 2004.
[4]	S. Balaji und D. M. S. Murugaiyan, „WATEERFALL Vs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC," *International Journal of Information Technology and Business Management,* Nr. Vol. 2 No. 1, 29 06 2012.
[5]	T. Gifford, „Are All Information Security Standards Equal?," [Online]. Available: https://www.optimisingit.co.uk/information-security-standards/. [Zugriff am 28 11 2017].
[6]	W. Jackson, „Under Attack," 10 08 2007. [Online]. Available: https://gcn.com/Articles/2007/08/10/Under-attack.aspx?Page=1. [Zugriff am 28 11 2017].
[7]	W. B. Arthur, „Inductive Reasoning and Bounded Reality," 1994. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.610&rep=rep1&type=pdf. [Zugriff am 24 01 2018].
[8]	Cambridge University Press, „Cambridge Dictionary," [Online]. Available: https://dictionary.cambridge.org/dictionary/english/assessment. [Zugriff am 22 11 2017].
[9]	Bibliographisches Institut GmbH, „Duden | Test," [Online]. Available: https://www.duden.de/rechtschreibung/Test. [Zugriff am 22 11 2017].
[10]	Internation Software Testing Qualifications Board, „ISTQB Glossary," [Online]. Available: http://glossary.istqb.org/. [Zugriff am 22 01 2018].
[11]	A. Tinkham und P. C. J. D. Kaner, „Learning Styles and Exploratory Testing," 2003. [Online]. Available: http://www.testingeducation.org/a/lset.pdf. [Zugriff am 22 01 2018].
[12]	Scrum Alliance,Inc, „Agile Atlas," 2012. [Online]. Available: https://improuv.com/file/193/download?token=p9twrIZR. [Zugriff am 23 11 2017].
[13]	K. Scarfone, W. Jansen und M. Tracy, „Confidentiality, Integrity, and Availability," Mozilla, 02 03 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Security/Information_Security_Basics/Confidentiality,_Integrity,_and_Availability. [Zugriff am 06 12 2017].
[14]	M. Nieles, K. Dempsey und V. Y. Pilitteri, „An Introduction to Information Security," NIST, 06 2017. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-12r1.pdf. [Zugriff am 08 01 2018].
[15]	C. F. Byrnes und P. E. Proctor, „Introduction to Security," in *The Secured Enterprise: Protecting Your Information Assets*, P. Hall, Hrsg., 2002.
[16]	H. Cavusoglu, B. Mishra und S. Raghunathan, „A Model for Evaluating IT Security Investments," *Communications of the ACM,* Nr. Vol. 47, No. 7, 07 2004.
[17]	Department of Defense, „Department of Defense standard practice: System safety," 11 05 2012. [Online]. Available: http://www.system-safety.org/Documents/MIL-STD-882E.pdf. [Zugriff am 05 12 2017].
[18]	J. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla und A. Murukan, „Threat Modeling," Microsoft Corporation, 06 2003. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa302419.aspx#c03618429_011. [Zugriff am 05 12 2017].
[19]	L. A. Gordon und M. P. Loeb, „The Economics of Information Security Investment," University of Maryland, 11 2002. [Online]. Available: https://www.researchgate.net/profile/Martin_Loeb/publication/220593665_The_Economics_of_Information_Security_Investment/links/57a1af2b08aeef8f311ccf1f/The-Economics-of-Information-Security-Investment.pdf. [Zugriff am 25 01 2018].
[20]	BITKOM, „ISO/IEC 15408 (CC)," [Online]. Available: http://www.kompass-sicherheitsstandards.de/43836.aspx. [Zugriff am 27 11 2017].
[21]	GCHQ, „Summary of risk methods and frameworks," 16 09 2016. [Online]. Available: https://www.ncsc.gov.uk/guidance/summary-risk-methods-and-frameworks. [Zugriff am 27 11 2017].

[22]	M. Waidner, M. Backes und J. Müller-Quade, „Entwicklung sicherer Software durch Security by Design," Frauenhofer Verlag, 05 2013. [Online]. Available: https://www.kastel.kit.edu/downloads/Entwicklung_sicherer_Software_durch_Security_by_Design.pdf. [Zugriff am 08 12 2017].
[23]	Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 27, IT Security techniques, „Common Criteria for Information Technology Security Evaluation," Beuth Verlag, Switzerland, 2014.
[24]	T. J. Smedinghoff, „The State of Information Security Law," Software Engineering Institute , 12 2007. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=52927. [Zugriff am 10 01 2018].
[25]	International Software Testing Qualifications Board, „Certified Testeer Advanced Level Syllabus - Security Tester," 18 03 2016. [Online]. Available: https://www.astqb.org/documents/Advanced-Security-Tester-Syllabus-GA-2016.pdf. [Zugriff am 15 01 2018].
[26]	Bitkom; Eito, „ITK-Marktzahlen," 10 2017. [Online]. Available: https://www.bitkom.org/NP-Marktdaten/ITK-Konjunktur/ITK-Markt-in-Deutschland/BITKOM-ITK-Marktzahlen-Oktober-2017-Kurzfassung.pdf. [Zugriff am 23 11 2017].
[27]	A. Berg und M. Hans-Georg, „Wirtschaftsschutz in der digitalen Welt," 21 07 2017. [Online]. Available: https://www.bitkom.org/Presse/Anhaenge-an-PIs/2017/07-Juli/Bitkom-Charts-Wirtschaftsschutz-in-der-digitalen-Welt-21-07-2017.pdf. [Zugriff am 23 11 2017].
[28]	J. Bird, „Bugs and Numbers: How many bugs do you have in your code?," 24 08 2011. [Online]. Available: http://swreflections.blogspot.de/2011/08/bugs-and-numbers-how-many-bugs-do-you.html. [Zugriff am 24 11 2017].
[29]	D. Meyer, „ratio of bugs per line of code," 11 11 2012. [Online]. Available: https://www.mayerdan.com/ruby/2012/11/11/bugs-per-line-of-code-ratio. [Zugriff am 24 11 2017].
[30]	A. Spillner, T. Linz und H. Schaefer, Software Testing Foundations, 4. Hrsg., Santa Barbara, CA: Rocky Nook, 2014.
[31]	Kaspersky Lab, „Lack of security talent: an unexpected threat to corporate cybersafety," 2016. [Online]. Available: https://kasperskycontenthub.com/press/files/2017/05/Kaspersky_Lab_Demand_for_Security_Talent_Report.pdf. [Zugriff am 30 11 2017].
[32]	R. Mack, Y. Ravin und R. J. Byrd, „Knowledge portals and the emerging digital workplace," IBM, 2001. [Online]. Available: http://old.disco.unimib.it/simone/TCA/portals.pdf. [Zugriff am 26 01 2018].
[33]	D. Tapscott, Grown Up Digital: How the Net Generation Is Changing Your World, Mcgraw-Hill, 2008.
[34]	University of Minnesota, „Human Resource Management," [Online]. Available: http://open.lib.umn.edu/humanresourcemanagement/chapter/8-5-cases-and-problems/. [Zugriff am 18 01 2018].
[35]	Cambridge Dictionary, „Meaning of "strategy" in the Enlish Dictionary," [Online]. Available: https://dictionary.cambridge.org/dictionary/english/strategy. [Zugriff am 15 01 2018].
[36]	M. Soni, „Defect Prevention: Reducing Costs and Enhancing Quality," [Online]. Available: https://www.isixsigma.com/industries/software-it/defect-prevention-reducing-costs-and-enhancing-quality/. [Zugriff am 27 11 2017].
[37]	J. Bach, „Test Strategy: What is it? What does it look like?," 1999. [Online]. Available: http://www.satisfice.com/presentations/strategy.pdf. [Zugriff am 27 11 2017].
[38]	C. W. Kirkwood, „Decision Tree Primer," Arizona State University, 2002. [Online]. Available: http://www.public.asu.edu/~kirkwood/DAStuff/decisiontrees/DecisionTreePrimer-1.pdf. [Zugriff am 06 12 2017].
[39]	J. Meier, C. Farre, J. Taylor, P. Bansode, S. Gregersen, M. Sundarajan und R. Boucher, „Threats and Countermeasures for Web Services," MSDN, 02 2009. [Online]. Available: https://msdn.microsoft.com/en-us/library/ff650168.aspx. [Zugriff am 06 12 2017].
[40]	C. Bornemann, Datenflussdiagramm, SPC TEIA Lehrbuch Verlag, 2002.
[41]	H. Röder, S. Franke, C. Müller und D. Przybylski, „Ein Kriterienkatalog zur Bewertung von Anforderungsspezifikationen," *Softwaretechnik-Trends,* Bd. 29:4, 11 2009.
[42]	M. E. Fagan, „Design and code inspections to reduce errors in program," *Software Pioneers,* pp. 575-607, 2002.

[43]	e. a. Berners-Lee, „Hypertext Transfer Protocol -- HTTP/1.0," Network Working Group, 05 1996. [Online]. Available: https://tools.ietf.org/html/rfc1945#section-11. [Zugriff am 18 12 2017].
[44]	Microsoft Corporation, „Microsoft Application Architecture Guide, 2nd Edition: Chapter 16: Quality Attributes," 10 2009. [Online]. Available: https://msdn.microsoft.com/en-us/library/ee658094.aspx. [Zugriff am 05 12 2017].
[45]	J. A. Nacif, F. M. de Paula, H. Foster, C. N. Coelho Jr., F. C. Sica, D. C. da Silva Jr. und A. O. Fernandes, „An Assertion Library for On-Chip White-Box Verification at Run-Time," Universidade Federal de Minas Gerais, Brazil; Mindspeed Technologies, USA; Universidade Federal de Minas Gerais, Brazil; Verplex Systems, USA; Universidade Federal de Ouro Preto, Brazil, [Online]. Available: http://www.cs.ubc.ca/~depaulfm/Papers/latw_03_final.pdf. [Zugriff am 15 12 2017].
[46]	J. Bergeron, Writing Testbenches: Functional Verification of HDL Models, Dordrecht: Kluwer Academic Publishers, 2002.
[47]	H. Tompson, „Why security testing is hard," in *IEEE Security & Privacy (Volume: 99, Issue: 4, July-Aug. 2003)*, 2003.
[48]	K. Scarfone, M. Souppaya, A. Cody und A. Orebaugh, „Technical Guide to Information Security Testing and Assessment - Recommendations of the National Institute of Standards and Technology," 09 2008. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf. [Zugriff am 16 11 2017].
[49]	R. Shapiro, S. Bratus, E. Rogers und S. Smith, „Identifying Vulnerabilities in SCADA Systems via Fuzz-Testing.," *Critical Infrastructure Protection V. ICCIP 2011. IFIP Advances in Information and Communication Technology,* Nr. 367, 2011.
[50]	Deja vu Security, LLC, „State Modeling," 23 02 2014. [Online]. Available: http://community.peachfuzzer.com/v3/StateModel.html. [Zugriff am 2017 12 05].
[51]	C. Smith, The car hacker's handbook: a guide for the penetration tester, W. Pollock, Hrsg., San Francisco, CA: No Starch Press, Inc., 2016.
[52]	Trusted Computing Group, „Trusted Platform Module Library, Part 1: Architecture," 29 09 2016. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.38.pdf. [Zugriff am 13 12 2017].
[53]	R. S. Sandhu, E. J. Coyne, H. L. Feinstein und C. E. Youman, „Role-Based Access Control Models," 26 10 1995. [Online]. Available: http://www.comp.nus.edu.sg/~tankl/cs5322/readings/rbac1.pdf. [Zugriff am 05 02 2018].
[54]	P. Loscocco und S. Smalley, „Integrating Flexible Support for Security Policies into the Linux Operating," USENIX Association, 06 2001. [Online]. Available: https://www.usenix.org/legacy/event/usenix01/freenix01/full_papers/loscocco/loscocco.pdf. [Zugriff am 05 02 2018].
[55]	M. Russinovich, „AccessChk," Microsoft Corporation, [Online]. Available: https://technet.microsoft.com/de-de/sysinternals/accesschk.aspx. [Zugriff am 05 02 2018].
[56]	M. Russinovich, „AccessEnum," Microsoft Corporation, [Online]. Available: https://technet.microsoft.com/de-de/sysinternals/bb897332. [Zugriff am 05 02 2018].
[57]	J. H. Saltzer und M. D. Schroeder, „The Protection of Information in Computer Systems," 17 04 1975. [Online]. Available: http://www.cs.virginia.edu/~evans/cs551/saltzer/. [Zugriff am 05 02 2018].
[58]	N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj und T. Zimmermann, „What Makes a Good Bug Report?," *Proceedings of the 16th International Symposium on Foundations of Software Engineering,* 09 2008.
[59]	I. J. 1/SC, „Software and systems engineering -- Software testing -- Part 3: Test documentation," ISO, 09 2013. [Online]. Available: https://www.iso.org/standard/56737.html. [Zugriff am 16 01 2018].
[60]	K. R. Reddi und Y. B. Moon, „Modelling engineering change management in a new product development supply chain," 04 05 2013. [Online]. Available: https://surface.syr.edu/cgi/viewcontent.cgi?article=1017&context=mae. [Zugriff am 17 01 2018].
[61]	A. R. M. Nordin und S. Suhailan, „Managing Software Change Request Process: Temporal Data Approach," 06 2009. [Online]. Available: https://www.researchgate.net/profile/Suhailan_Safei/publication/41822929_Managing_Software_Change_Request_Process_Temporal_Data_Approach/links/568c8fac08ae71d5cd04dc2e.pdf. [Zugriff am 17 01 2018].

[62]	Common Criteria Recognition Arrangement, „Members of the CCRA," [Online]. Available: http://www.commoncriteriaportal.org/ccra/members/#DE. [Zugriff am 21 12 2017].
[63]	J. Dorschel, „IT-Sicherheit und Haftung – ein Überblick," *it-sicherheit,* pp. 58-59, 06 2009.
[64]	bitkom, „Wozu brauchen wir Standards?," [Online]. Available: http://www.kompass-sicherheitsstandards.de/43517.aspx. [Zugriff am 14 12 2017].
[65]	S. Northcutt, J. Shenk, D. Shackleford, T. Rosenberg, R. Siles und S. Mancini, „Penetration Testing: Assessing Your Overall Security Before Attackers Do," SANS Institute, 06 2006. [Online]. Available: https://www.sans.org/reading-room/whitepapers/analyst/penetration-testing-assessing-security-attackers-34635. [Zugriff am 21 12 2017].
[66]	Springer Gabler, „Gabler Wirtschaftslexikon," [Online]. Available: http://wirtschaftslexikon.gabler.de/. [Zugriff am 27 01 2018].
[67]	ISO/TC 22/SC 32, „Road vehicles -- Functional safety -- Part 10: Guideline on ISO 26262," 2012. [Online]. Available: https://www.iso.org/standard/54591.html. [Zugriff am 04 01 2018].
[68]	M. Amoozadeh, A. Raghuramu, C.-N. Chuah, D. Ghosal, H. M. Zhang, J. Rowe und K. Levitt, „Security Vulnerabilities of Connected Vehicles Streams and their Impact on Cooperative Driving," 04 2015. [Online]. Available: https://pdfs.semanticscholar.org/0965/69cb26b8bdefb9de914d8ca68670096e335f.pdf. [Zugriff am 22 01 2018].

9.4 FIGURE INDEX

9.5 TABLE INDEX

Diese Arbeit wurde von mir selbständig verfasst und in gleicher oder ähnlicher Fassung noch nicht in einem anderen Studiengang als Prüfungsleistung vorgelegt. Ich habe keine anderen als die angegebenen Hilfsmittel und Quellen, einschließlich der angegebenen oder beschriebenen Software, verwendet.

Stuttgart, den