Fraunhofer
SIT

FRAUNHOFER INSTITUTE FOR SECURE INFORMATION TECHNOLOGY
SIT TECHNICAL REPORTS

Trends and Strategy Report

Development of Secure Software with Security By Design

Michael Waidner, Michael Backes, Jörn Müller-Quade

FRAUNHOFER VERLAG

CISPA
Center for IT-Security, Privacy

EC SPRIDE
EUROPEAN CENTER FOR
SECURITY AND PRIVACY BY DESIGN

KASTEL

Competence Centers for Cyber Security

Fraunhofer
SIT

Development of Secure Software with Security by Design

Michael Waidner (Hrsg.), Michael Backes (Hrsg.), Jörn Müller-Quade (Hrsg.), Eric Bodden, Markus Schneider, Michael Kreutzer, Mira Mezini, Christian Hammer, Andreas Zeller, Dirk Achenbach, Matthias Huber, Daniel Kraschewski

SIT Technical Reports
SIT-TR-2014-03

July 2014

Fraunhofer Institute for Secure
Information Technology SIT
Rheinstraße 75
64295 Darmstadt
Germany

SPONSORED BY THE

This trends and strategy report has been funded by the German Federal Ministry of Education and Research.

Federal Ministry
of Education
and Research

FRAUNHOFER VERLAG

IMPRINT

Contact
Fraunhofer Institute for
Secure Information Technology SIT
Rheinstra e 75
64295 Darmstadt
Germany

Phone	+49 (0) 6151 869-213
Fax	+49 (0) 6151 869-224
E-Mail	info@sit.fraunhofer.de
URL	www.sit.fraunhofer.de

Ed. Michael Waidner
SIT Technical Reports
SIT-TR-2014-03: Development of Secure Software with Security by Design
Michael Waidner (Hrsg.), Michael Backes (Hrsg.), Jörn Müller-Quade (Hrsg.), Eric Bodden, Markus Schneider, Michael Kreutzer, Mira Mezini, Christian Hammer, Andreas Zeller, Dirk Achenbach, Matthias Huber, Daniel Kraschewski
ISBN 978-3-8396-0768-8
ISSN 2192-8169

Printing:
Mediendienstleistungen des
Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart

Printed on acid-free and chlorine-free bleached paper.

Development of Secure Software with Security by Design

Michael Waidner (Hrsg.), Michael Backes (Hrsg.), Jörn Müller-Quade (Hrsg.), Eric Bodden, Markus Schneider, Michael Kreutzer, Mira Mezini, Christian Hammer, Andreas Zeller, Dirk Achenbach, Matthias Huber, Daniel Kraschewski

This trends and strategy report argues that the development and integration of secure software has to follow the *Security by Design* principle and defines respective challenges for a practice oriented research agenda. Software is the most important driver for innovations in many industries today and will remain so in the future. Many vulnerabilities and attacks are due to security weaknesses in application software. During application software development or integration, security issues are either taken into account insufficiently or not at all, which constantly leads to new openings for attacks. Besides functionality, software security is becoming more important to users and manufacturers. Using new practical methods and systematically observing security processes will help software managers and integrators to avoid security vulnerabilities. Development and security processes improvement offers manufacturers the opportunity to reduce software costs and development time whilst gaining enhanced security features. For companies this step is very important strategically and highly relevant for their medium to long-term competitiveness. Since software products and their development processes can be very complex, it is not clear to manufacturers how they can profit from and economically realize *Security by Design* and the security processes necessary for it. It is the purpose of applied research to address the challenges within this context, and to master them and transfer realizable solutions into practical use.

Key Words: Security by Design, Secure Engineering, Software Engineering, Security Development Lifecycle, Application Security, Supply Chain, Software Development

Michael Waidner (Hrsg.)
EC SPRIDE, TU Darmstadt,
Fraunhofer-Institut für Sichere Informationstechnologie (SIT)
Fraunhofer SIT, Rheinstra e 75, 64295 Darmstadt
`www.sit.fraunhofer.de`, `www.ec-spride.de`, `www.informatik.tu-darmstadt.de`

Michael Backes (Hrsg.)
CISPA, Saarland University
Universität des Saarlandes, Postfach 151150, 66041 Saarbrücken
`www.cs.uni-saarland.de`, `www.cispa-security.de`

Jörn Müller-Quade (Hrsg.)
KASTEL, Karlsruher Institut für Technologie (KIT)
Karlsruher Institut für Technologie, Kaiserstra e 12, 76131 Karlsruhe
`www.kit.edu`, `www.kastel.kit.edu`

Eric Bodden, Markus Schneider
EC SPRIDE, Fraunhofer SIT
Fraunhofer SIT, Rheinstra e 75, 64295 Darmstadt
`www.ec-spride.de`, `www.sit.fraunhofer.de`

Michael Kreutzer, Mira Mezini
EC SPRIDE, TU Darmstadt
EC SPRIDE, Mornewegstra e 30, 64293 Darmstadt
`www.ec-spride.de`, `www.informatik.tu-darmstadt.de`

Christian Hammer, Andreas Zeller
CISPA, Universität des Saarlandes
Universität des Saarlandes, Postfach 151150, 66041 Saarbrücken
`www.cispa-security.de`, `www.cs.uni-saarland.de`

Dirk Achenbach, Matthias Huber, Daniel Kraschewski
KASTEL, Karlsruher Institut für Technologie (KIT)
Karlsruher Institut für Technologie, Kaiserstra e 12, 76131 Karlsruhe
`www.kastel.kit.edu`, `www.kit.edu`

CONTENTS

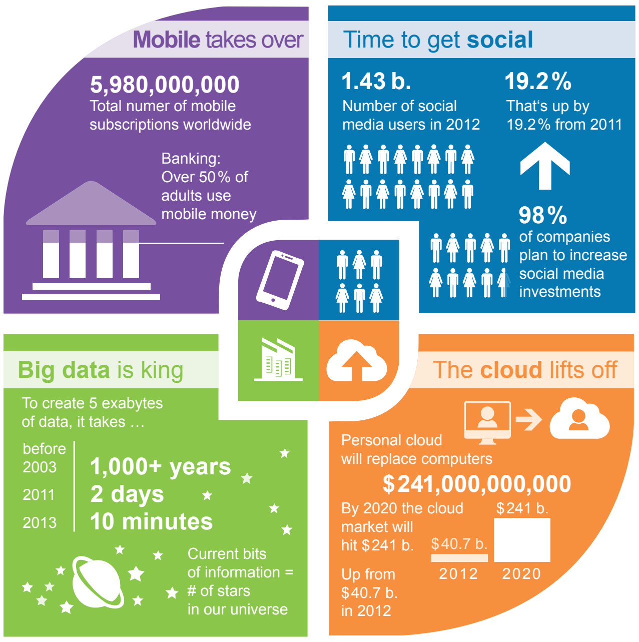
Greeting by Karl-Heinz Streibich

Software AG, Chief Executive O cer

Dear Ladies and Gentlemen,
Esteemed Colleagues from Science and Industry,

Today the digital world is generating every two days as much data as has been in use in the time between the beginning of human civilization and the year 2003. Billions of mobile end devices are being used. We cannot fathom our everyday life without them anymore: users document where they are, who they are talking to, what moves them. The classic mobile phone has turned into a source of data.

For the first time we are equipped with the technical possibility to survey our environment, our daily routine and our life in realtime. In the global software industry this is a unique constellation, because four technological megatrends meet at the same time:



- Mobile   the increasing mobile communication and the mobile Internet use.
- Cloud Computing   the transfer of data and applications into the internet.
- Social Collaboration   the increased usage of social networks.
- Big Data   the processing and analysis of vast data amounts in realtime.

Software has become the fundamental material and innovation driver in nearly all industries. Processes, products and production methods are being connected over the Internet and can be augmented with digital information and interlinked in a whole new manner. With this increasing interconnection the customer s need for secure, digital solutions over the entire value chain grows as well. Today Software

AG is leading in 15 market sectors with its product families Adabas and Natural, webMethods, ARIS and Terracotta. We offer our customers the qualitatively best solutions for digitalizing their enterprise. Our leading market position is the result of decades of research and developmental work, even beyond company boundaries, and the foundation for the strategic partnership with the European Center for Security and Privacy by Design (EC SPRIDE). This partnership allows Software AG to benefit from a scientific institute s high-level in the area of IT security and integrate the results into its own software development process. The focus of the joint activities is on the Laboratory for Secure Engineering. This Secure Engineering Lab forms the organizational framework for the joint research activities, the expansion of our development team and the continued optimization of our joint development processes based on the latest research results. The software production methods have to adapt to the new requirements and conditions that are characterized more by the decentralization and distribution of development works (worldwide distributed development teams, the integration of third party and open source components and cross-company processes). Security needs to be included into the development process from the very beginning (Security by Design), which makes it a must that IT tools will have to be modified and augmented. EC SPRIDE and Software AG work together in these areas to put the latest research findings into practice, given the respective specific conditions.

The goal is to intermesh industry and science, because in the future innovative products and services are not conceivable anymore without secure software. The German industry s competitiveness will be determined by its capability to create software based products and services of utmost quality. Software competency will be the prerequisite for Germany to maintain its leading position in engineering and further expand its position as one of the leading export nations. A dynamic and successful German software industry gives an important impetus to all types of economic sectors and thus for the competitiveness of the German national economy. This is why the cooperation with an active and dedicated research community such as EC SPRIDE is of important concern to us.

Yours,

[illegible]

Karl-Heinz Streibich - Vorstandsvorsitzender der Software AG

1. THE CHANGING FACE OF SOFTWARE SECURITY AND SOFTWARE DEVELOPMENT

These days most innovations are based on information technology. This is true for IT sector innovations, as well as for other sectors such as energy, finances, health, trade, logistics, media, production, environment and tra c. Information technology, which is frequently implemented as software, plays a prominent role in this.

Nowadays companies and organizations are employing application software in important business processes that are frequently vital for the business success. Such application software features special functions needed for the most varied purposes. Today application software development considers these desired functions almost exclusively. The developers are experts in their respective application domains. During the developmental process, security is taken into consideration either only marginally or not at all. Inevitably, this leads to security vulnerabilities in the application software. Consequentially hackers repeatedly try to gain access to data and systems via these weaknesses in security in order to profit at the expense of others [BKA12; BKA11]. Besides the functionality of the software itself, software security is becoming more important to users and manufacturers. Security vulnerabilities in application software constitute a big risk for organizations and enterprises and they are now understood to be the most dangerous source of threats (see for example figure 1). For users, the realistic concern of financial losses puts the security issue increasingly at the center. Software application manufacturers are called upon to react accordingly and to improve the security of their products.

In the past manufacturers tried to externalize the security tasks. This was done with firewalls, wrappers, intrusion detection or malware protection. If application software has security vulnerabilities, it may not always be possible to remedy these weaknesses via externally added security components without a loss in functionality. This currently widespread software development practice leads to constant detection of vulnerabilities, which then have to be dealt with as fast as possible with elaborate and expensive patch cycles.

Since the causes for security vulnerabilities are understood better in practice now than they were a few years ago, awareness is rising that software security should be factored in more extensively during development and integration . The threat and risk situation will not progress substantially unless application software security is improved.

To improve application software security it is urgently required that security is factored in from the very beginning of software development, i.e. during the design phase, and tracked throughout the full software development lifecycle (see for example *Security Development Lifecycle* (SDL) of Microsoft [Mic10]). Manufacturers anticipate from this approach both products with better security features and a reduction in manufacturing costs [For11a; Abe10]. The earlier such a security process

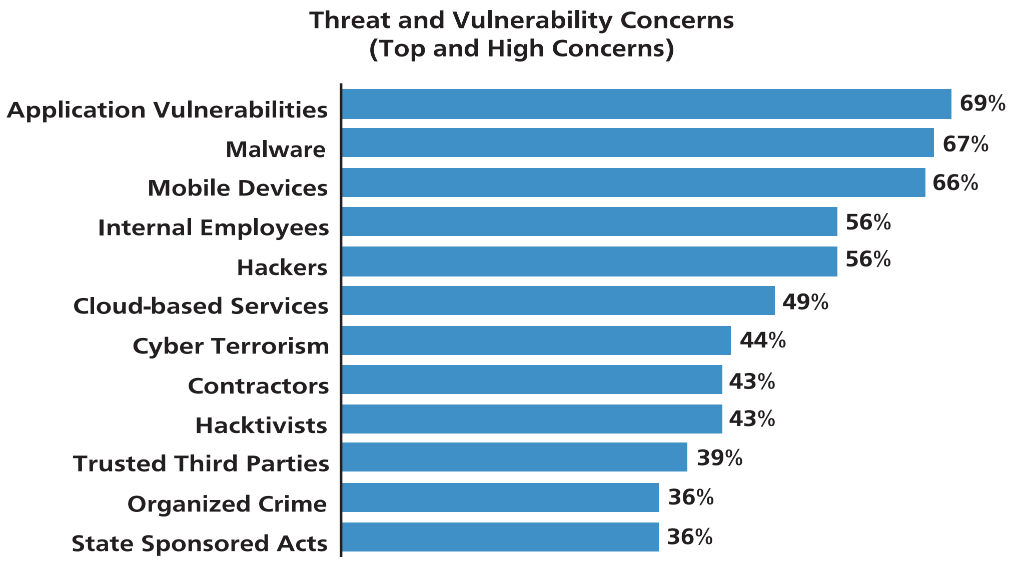


Figure 1: According to a study by Frost & Sullivan, (ISC)^2 and Booz, Allen, Hamilton represent security vulnerabilities in application software the biggest threat. (source: [FIB13])

detects vulnerabilities during the development, the lower the costs for a remedy: Implementing security measures after the fact is significantly more expensive and usually offers less protection than security that was integrated into the system development process or into the product selection process from the very beginning. Security should therefore be an integrated component of an IT system s or product s entire lifecycle. [BSI06]

This clearly demonstrates the strategic dimension of security processes. If software manufacturers adapt and improve their development and security processes accordingly they can improve their products security as well as their competitiveness. This requires a paradigm shift so that security processes can be realized in an economical manner and where the individual enterprises are willing to fund the initial investments for this change. The adoption of security processes is an important aspect in order for software manufacturers to prevail against competition.

Software and software development processes can be very complex, especially with larger projects. For example, one single modern software end product may contain software components from many different manufacturers, for which the current security processes are insu cient. For economic reasons, and to save time, components that have been previously developed under different criteria may be integrated (legacy). The complexity of software development and the *human* factor in development repeatedly brings about errors and thus weaknesses. These problems are mitigated by using supporting tools.

In view of the need for secure software on one hand and the vulnerability of industry and society on the other, the security processes during software development must undergo extreme changes. However, during software manufacture and integration, transformation processes will be successful only if they can be designed evolutionarily. It must be kept in mind that manufacturers will not be able to resort to other developer resources *ad hoc*. In industrial software development it is therefore very important to design new tools which will have to be integrated into existing developmental environments, and which support current developers, whose security expertise is less pronounced, in preventing security vulnerabilities. It is expected that industrial software development and the accompanying security processes will evolve immensely in the coming years. In the end it is expected that software security will be taken into consideration during the design phase and that the security will be improved systematically and methodically over the software lifecycle. This expectation is characterized by various forecasts describing the ideal development of secure software from differing perspectives. Research has to address and overcome a number of challenges in order for these ideas to become reality. In the next step, the results then have to be transferred into real software development.

This trends and strategy report describes the ideal of future secure software development and outlines the challenges which will determine the practice oriented research agenda in the years to come.

2. THE SIGNIFICANCE OF SECURITY BY DESIGN

2.1 The Term Security by Design

The term *Security by Design* may be understood in different ways. In the more narrow sense *Security by Design* means considering security as early as the design phase of the software development process. In a broader sense *Security by Design* can be understood as the systematically organized and methodically equipped framework that is applied over the lifecycle of secure software. For example, this framework may include the embedding of secure software development at governance level, individual security processes for the software s lifecycle phases, and security analyses of software components integrated from other manufacturers. In this document, *Security by Design* is to be understood in the broader sense.

2.2 The signi cance for Society

Software is very important for society and the functioning and maintenance of our social system, and secure software is particularly so. Modern information technology and software have found their way into almost all areas of daily life, with state institutions, companies or with private users. The significance of *Security by Design* for society is illustrated by the following points:

— Prosperity: Information technology today contributes in many respects to citizens prosperity. As the major innovation and productivity driver, information technology is saving peoples jobs, and is thus the basis for the prosperity of these people. In Germany the digital economy and its net product has already surpassed the German traditional industries such as the automobile industry and mechanical engineering [BMW12b; BMW12a]. Information technology and the internet have become our society s backbone and nervous system. Even the way citizens interact as social beings is heavily shaped by information technology and thus by software. In communication and other everyday information processes, for example during shopping or information research, software frequently plays an important role. In all of these applications and contexts it is important for the citizens that they are protected. Time and time again security vulnerabilities come to light which pose a considerable risk for many citizens, even if the technology used for this has already existed in principle for more than 10 years. For example the breach found in 2013 at Amazon [hei13] or the Sony PlayStation breach, in which the data of more than 70 million customers was stolen [hei11]. More and more citizens are becoming afraid of security breaches and attacks [hei12b]. *Security by Design* with specific attention to the security processes improved during the manufacture of application software can reduce the risks for society.

– Economy: The benefit that the German economy can derive from secure software and *Security by Design* has a social dimension. As a high-wage country Germany has to depend on the realization of innovative ideas, the quality of its products, and production processes that can be designed e ciently and economically. In an open, connected and digitalized world companies depend on protecting their knowledge, which represents the basis of their competitive advantage, against competitors and potential attackers. *Security by Design* provides the economy s stakeholders with an improved starting position for protecting their own interests. If this is to succeed, the position of particularly the small and medium-sized enterprises has to be improved. Today small and medium-sized software manufacturers are no longer in a position to hone their development processes on their own. For this, applied research has to prepare the groundwork and give the required support.
– eGovernment: Software is indispensable for government institutions. This is true for both internal processes as well as for executing processes with citizens. This includes many processes where the need for secure software is obvious, for example when filing taxes electronically with the tax authorities. Significant risks obviously exist with regard to the security of software used in public o ces [WAZ12]. *Security by Design* helps to improve software security for eGoverment.
– Public security: Public security comprises a nation s inner and outer security. Entities that are active within this context, for example the police, frequently have to rely on modern information technology to organize and execute their work. Since organized crime and international terrorism have changed the threat situation immensely (e. g. by using modern information technology), the government representatives have to tackle new tasks in order to be able to reduce the risks for society [RGWS08]. To reduce risks and targets it is important to reduce the vulnerabilities in software used by government bodies.
– Critical infrastructures: In critical infrastructures such as power, communication networks, water supply, or transport, information technology is used to a substantial extent. In view of the great significance these infrastructures have for society it is very important that the software used in the infrastructures is secure against attacks, e.g. during manipulations or acts of sabotage. To reduce the infrastructure vulnerability the software used should be secure and therefore have been developed according to the *Security by Design* paradigm. The German federal government s plan to pass an IT security law placing the critical infrastructure operators under the obligation to provide more IT security is a step in the correct direction.
– Democracy: The Arab Spring (see e.g. [Nü12]) demonstrated that information technology can contribute to democratization processes. However, information technology is important for the democracies in Europe as well: It helps to organize processes that are indispensable in a democracy. For example, information that is required for citizens to form an informed opinion can be procured

quickly and practically at no cost. Other important processes such as debates and exchange with others are becoming easily possible by overcoming space and time constraints. Information technology and interconnection can provide transparency and serve the people in evaluating politics and government entities. In a democracy these processes require citizens to be self-determined and free. In this context data protection and software security play an essential part. *Security by Design* supports this.

2.3 The Signi cance for Software Users

Users need software with excellent features. This is true for both professional and private use. Security vulnerabilities in software can represent a high risk for users, especially if the software is used in areas that are critical for business success, associated with real financial losses or may threaten one s existence. The following examples will demonstrate the unpleasant ramifications of security vulnerabilities:

- By exploiting security vulnerabilities, an infiltrated malcode spied on and pillaged Nortel over years [Spi12]. For years the problem was not taken seriously. The attackers had "access to everything", said Brain Shields, the manager who headed the inquiry at the time [hei12a]. A multitude of possibilities for attacks opens up if attackers succeed in implementing a malcode. Once an attacker has accomplished this there is very little that can be done in defense by *Security by Design.* But *Security by Design* can help to make malcode installation much more di cult for attackers.
- The New York Times was spied on as well by e-mails distributing malcode on the staffs computers [Spi13]. It is believed that the attacks were intended to uncover the identity of those informants that collaborated with the newspaper s journalists.
- In 2012, hackers managed to rob a total of over 36 million Euros from more than 30,000 bank customers using the malsoftware Eurograbber which targets online banking [DMN12].

If the *Security by Design* paradigm is used in software development, many security vulnerabilities can be avoided, which in turn reduces the risks for users. Besides direct losses security vulnerabilities may result in additional problems for users. The loss of reputation can be one of them. In enterprises the question of liability arises, for example towards customers or partners that suffer from a disadvantage due to the user s security vulnerabilities. It is also conceivable that high-level decision makers may be held liable, for example if software containing security vulnerabilities was used this may be considered as negligence.

Reducing security vulnerabilities by *Security by Design* reduces in turn the expenditures for maintenance processes on the user side, because security patches have to be acquired, tested and possibly distributed and installed much less frequently.

This reduces the operative software costs (*cost of ownership*). Besides, it cannot be assumed in all cases that users have the expert knowledge to assess adequately the risks posed by certain security vulnerabilities. Improving the initial situation for users through *Security by Design* carries a psychological component as well, because anxieties concerning the use of modern technology may be discarded or minimized and a trusting interaction with technology may be encouraged instead.

Especially users whose software costs represent a major proportion of their budget are increasingly starting to question manufacturers about the security processes applied with *Security by Design* and often request them to demonstrate their methods for software with enhanced security features. The mere existence of such security processes may be a significant criterion for a user when deciding about a software purchase. However, knowing that product manufacturing processes comply with the *Security by Design* paradigm may be of interest to users with little market power, as well as for private individuals. Such information may be helpful particularly for those users that are less familiar with IT security issues. The restructuring of production processes turned out to be a market success in other areas, for example with wholefood products.

2.4 Signi cance for Software Manufacturers

For enterprises, the introduction of *Security by Design* may have existential implications. A number of reasons speak for introducing this paradigm into current production processes, among them:

— Reducing secure software development costs: This becomes clearer when regarding current software development and security processes. In the past security frequently played only a minor role if any at all. Often security experts were not involved until a product s development was quite advanced. If the experts found vulnerabilities, the chosen architecture and design did not always permit closing these weaknesses in a simple and easy way. To eliminate such vulnerabilities major changes had to be done on parts of the respective software, if it was possible at all. This process would destroy achievements which were often funded specifically at the beginning of the development. Such situations can be avoided by considering the security requirements as early as during the software s design phase. When compared to the traditional approach the cost saving potential is greater the earlier corrections can be carried out. This realization is hardly new. More than 10 years ago NIST compared the costs for remediating unless this is an IT term I am unfamiliar with, the correct word is remedying bugs during the various phases [Tas02]. One of the results from the study is illustrated in figure 2. It shows that the average costs of eliminating bugs early compared with those of eliminating them later vary by a factor of 30. It can be assumed that the factor of this disproportion may be even higher when considering solely the security vulnerabilities. This assessment is confirmed by the data shown in

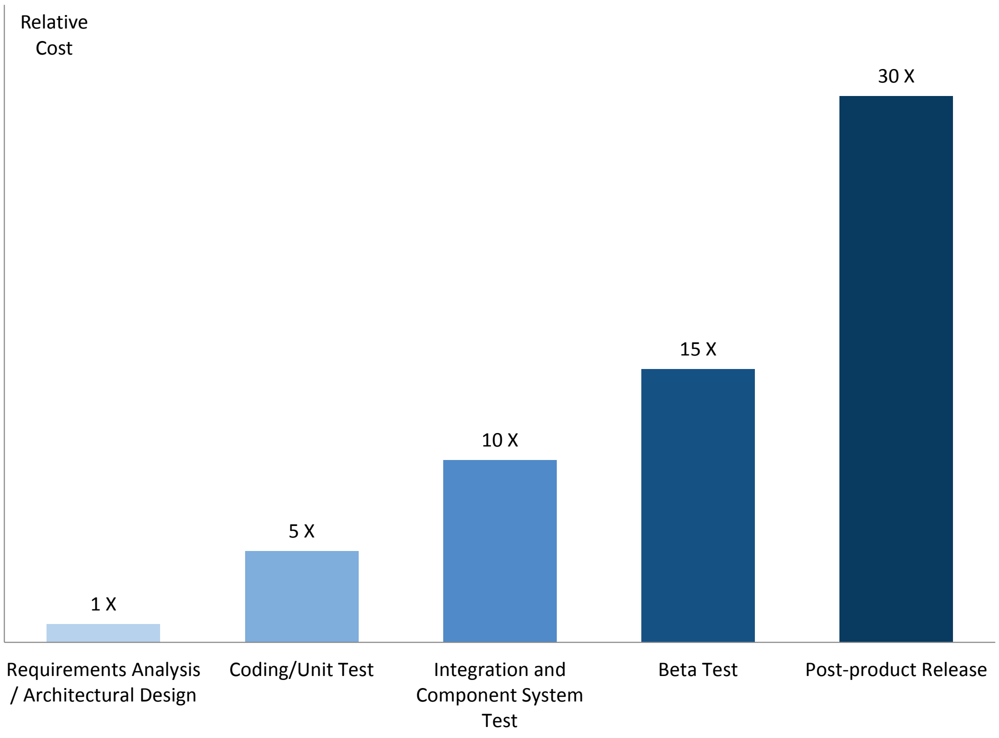
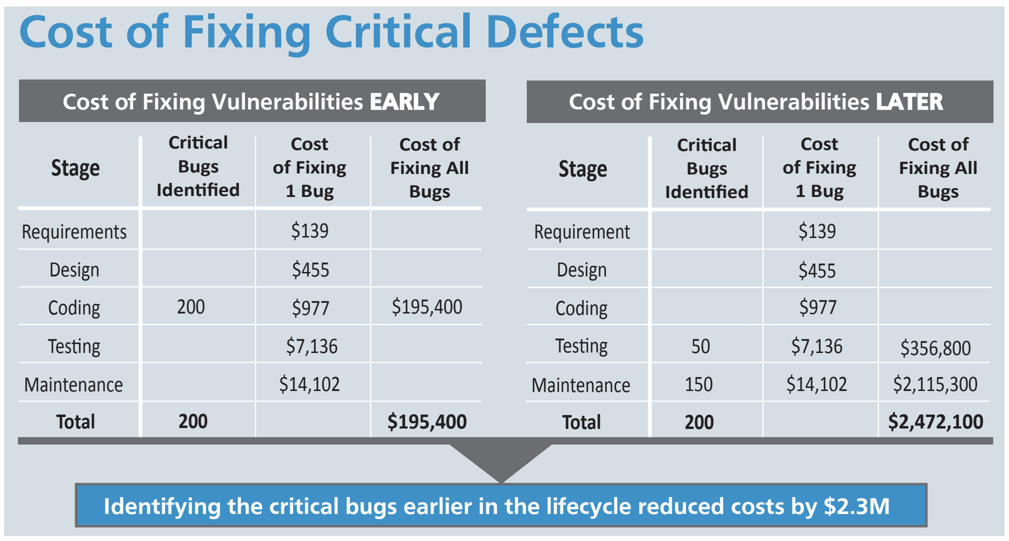


Figure 2: The cost development in relation to remediating bugs during the various phases of a software lifecycle according to a NIST study (source: [Tas02]).



Cost of Fixing Critical Defects

Cost of Fixing Vulnerabilities EARLY

Stage	Critical Bugs Identified	Cost of Fixing 1 Bug	Cost of Fixing All Bugs
Requirements		$139	
Design		$455	
Coding	200	$977	$195,400
Testing		$7,136	
Maintenance		$14,102	
Total	**200**		**$195,400**

Cost of Fixing Vulnerabilities LATER

Stage	Critical Bugs Identified	Cost of Fixing 1 Bug	Cost of Fixing All Bugs
Requirement		$139	
Design		$455	
Coding		$977	
Testing	50	$7,136	$356,800
Maintenance	150	$14,102	$2,115,300
Total	**200**		**$2,472,100**

Identifying the critical bugs earlier in the lifecycle reduced costs by $2.3M

Figure 3: The di erent costs for remediating critical bugs during the various phases (source: [VK11]).

[VK11] (see also figure 3): The average costs for eliminating critical bugs between the phases *requirements* and *maintenance* add up to a factor larger than 100.

– Improving software security: The systematic use of security processes accords security during the development process a much higher significance than it had received before. Security issues are taken into consideration and analyzed over the full lifecycle. This results in the improvement of software security. This is demonstrated for example by Microsoft and *SDL* [Mic13b]. Figure 4 shows an example where the security features of two Microsoft products were improved after *SDL* had been implemented. Another example is the implementation of the *Adobe Secure Product Lifecycle* (SPLC) [Ado13]: It lead to significantly improved quality and higher resistance against attacks in the products *Adobe Reader* and *Adobe Flash*.
– Reducing the costs for patch provisioning: Improving the security features reduces the number of vulnerabilities. As an immediate consequence this reduces the frequency with which security updates or patches are required. Further down the line it reduces the costs the manufacturers have to bear for patch development, tests, provisioning and support.
– Maintaining the manufacturer s reputation: By improving his proprietary products security features a manufacturer will receive less negative attention in the media due to security vulnerabilities. Manufacturers can use the realization of the *Security by Design* paradigm in a positive sense. Customers highly appreciate investments made to improve production processes for the consumers benefit.
– No market limitation: A production process that does not follow the norm may be a criterion for exclusion in a customer s decision between manufacturers or products. In light of this it is important for manufacturers to realize *Security by Design* in order not to limit their own sales markets.
– Improving competitiveness: The decision to realize *Security by Design* in one s own production processes at the right time improves competitiveness. Such an improvement, however, can only be achieved, if the realization is not too late when compared with the most important competitors, as otherwise market shares may be lost. Regaining market shares may be very di cult, because customers may not be won back immediately once they have opted for a competitor s product.

For manufacturers the restructuring of development processes to follow *Security by Design* is a strategic decision with far-reaching medium- to long-term consequences. This decision has to be implemented company-wide and requires certain investments during the implementation phase, which will amortize once a routine has been established in these processes.

Many manufacturers are medium-sized companies and cannot accomplish software development process restructuring to *Security by Design* on their own. Only global corporations of a size such as Microsoft or IBM can accomplish such transformation processes in their production on their own. For manufacturers not quite as large as those it is important that they are supported in their implementation of *Security*

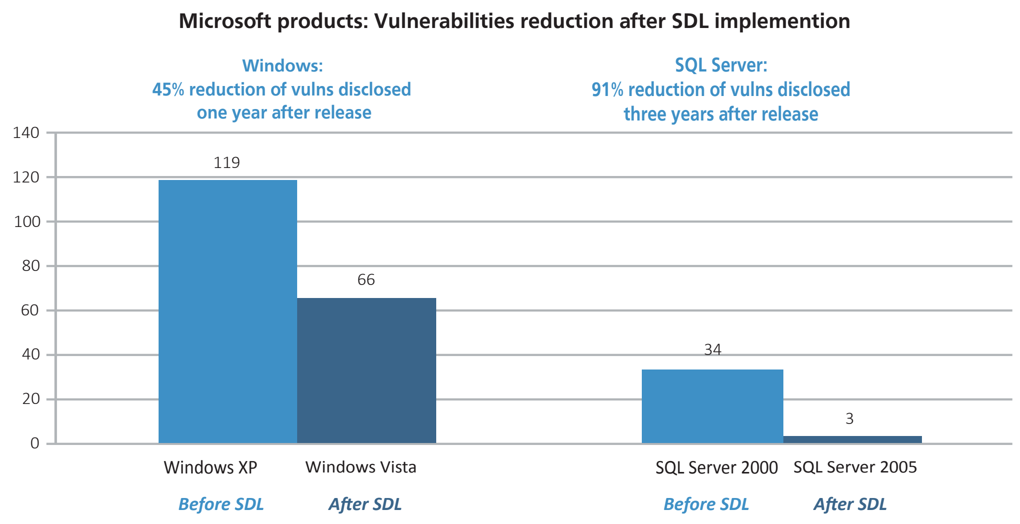


Figure 4: The impact of SDL on the security of software (source: [Mic13b]).

by Design approaches. This allows smaller manufacturers to remain competitive in their niches when compared with the larger manufacturers.

For the practical implementation of *Security by Design* it is absolutely essential that research considers today s established peculiarities and characteristics of software production processes. Production processes may be very complex and can be defined by many constraints such as:

— time pressure
— cost effectiveness
— pressure to be innovative
— compliance demands for certain industries or countries
— lines of products
— integration of supplier codes
— integration of open source components
— legacy code use
— reducing human error in uences
— measurability and controlability of measures within *Security by Design*

Introducing new methods and security processes in software manufacturing and implementation has to be manageable and controllable. The effects of individual measures during the transformation of the manufacturing processes have to be measureable in an objective way in order to be able to assess which measures are beneficial and realizable in a cost e cient way and which ones may need further modification. Each innovation within the *Security by Design* process at production level basically requires a corresponding solution at management level which enables checkability and controllability. The solution at management level has to consolidate the relevant security aspects with the information related to the constraints mentioned above, evaluate them and present them for decision support.

In addition to the approach based on *Security by Design*, manufacturers also have other options to improve the security of their production processes and products, for example by certifications such as *Common Criteria*. Even though this possibility has existed for many years, manufacturers usually avoid it for various reasons. Certification is expensive, time consuming and has to be repeated for even the smallest modification and further development of a product. Today certification is used mainly only for niche products with specific security requirements.

3. SOFTWARE SECURITY THROUGH AUTOMATION AND REDUCTION HUMAN FACTORS

The IBM X-Force reports [IBM12], the BSI progress reports [BSI13], the annual Coverity Scan Reports [Cov13] and the frequent computer bugs classified by SANS as dangerous [Chr11] all demonstrate concordantly in their analysis and evaluation that primarily the same vulnerability types have occurred for years. This means that the bugs and vulnerabilities resulting from this could have been avoided. For example, in his standard work about secure software development Gary McGraw establishes an entire taxonomy for such known potentially security relevant coding errors during the programming phase (*Coding Errors*) (compare chapter 12 in [McG06]). Mostly they are errors caused by the *human* factor. To understand how these errors originate from the *human* factor, it is helpful to take a look at the conditions under which software, in particular application software, is developed today. Even today software development is in many cases driven exclusively by the software s functionality. Security tends to play only a minor part, if any at all. The developers are experts in the respective software application domain; they do not give a high priority to security issues. The pressure for innovations affects the development of new functions and leaves developers only a small amount of room to deal with security issues as well. If security guidelines for software development actually exist, for example programming guidelines and manuals, they are often realized inadequately. Instead the degrees of variances in programming languages were often used without much thought, if a desired function could be achieved with it. When security aspects received a systematic consideration, they were typically externalized, for example by security experts developing specific security components such as wrappers, firewalls or virus scanners. Usually developers did not use already existing tools to detect software vulnerabilities.

Security vulnerabilities that have developed due to the *human* factor unfortunately cannot be changed e ciently and effectively in practice by combatting the causes, for example in uencing developers to convince them to modify their working methods. It is to be expected that human errors that can be traced back to ignorance, carelessness or   ightiness will occur to almost the same extent as before. The idea that a manufacturer may be able to change a large number of developers within a short time is unrealistic. One possibility to improve the situation is to provide the developers with technical solutions that keep them from committing the respective errors.

These man-made and by now well-known security errors could largely be avoided by assistance systems during development [Zel07; BBMM10] and by security oriented parameters. Once integrated into the development environment these assistance systems could automatically detect errors that lead to security problems and suggest alternatives to resolve them. Further evolution could even result in certain mistakes not being made anymore. Most of the remaining vulnerabilities could be detected with automatic or semi-automatic support during software roll-out. A vision can be formed from combining these points:

> *The software development process of the future will be defined by consistently security oriented programming languages and tools that can be integrated seamlessly. This will prevent security relevant errors as per the research's current state, and vulnerability detection will be systematic and mostly automated.*

Moreover, this evolution of the software development process will improve cost effectiveness in software development.

3.1 Challenge: Security Oriented Programming Languages Constructs, and Managed Code

Buffer over ows demonstrate impressively the problem of insu ciently security oriented programming languages. Buffer over ows have been exploited as security vulnerabilities for more than two decades and just as long and without interruption they have been part of the 25 most dangerous vulnerabilities [Chr11]. Basically, this concerns every code written in programming languages that do not automatically monitor storage area access prominent examples for these programming languages are C and C++.

Buffer over ows cannot occur in Java, if *Java Virtual Machine* (JVM) has been implemented correctly, because JVM controls storage area compliance. It is still possible, however, to invoke native code from various Java technologies so that buffer over ows are possible through the backdoor with Java programs as well.

Just like JVM, the managed code of the Microsoft .NET framework was designed with security in mind: bytecode carried out in *Common Language Runtime* (CLR) prevents vulnerabilities such as buffer over ow and privilege escalation. Unfortunately, the .NET framework s programming language C# is not consistent in preventing vulnerability-induced pointer arithmetic: The key word *unsafe* still permits pointer arithmetic.

Current *Java API* exploits turned the attention increasingly towards the Java security model: On April 17, 2013 Java 7 published a patch release with update 21, providing 42 patches for security errors, several of which reach the maximum value of 10 in the *Common Vulnerability Scoring System.* This is even more serious since these vulnerabilities exist for different operating systems due to Java s platform independence. These attacks utilize vulnerabilities during the securing of critical *Java API* resources such as *class loading* or *reflection* that developers introduced unknowingly during platform enhancement. Since the Java security model allows the active limitation of privileges, these vulnerabilities remain unnoticed and are adopted unnoticed into the new Java releases. A different model providing for security from the very beginning would render these vulnerabilities unusable or at least visible.

Type systems could be used much more widely than they are today: Type systems check and protect the semantics and thus represent an approach that achieves IT security by *safety*. The security models of managed code languages such as Java are inconceivable without a type system. For example, the Java type system ensures that pointer arithmetic remains impossible even if type conversions occur. Other parts of the security architecture rely on these invariants guaranteed by the type system. Type systems can be designed to be as powerful as desired. Approaches have been developed that in part far exceed type systems such as Java: For *Bali*, a variation of Java, a full security type system was introduced [ON98]. With [Loc12] a type-safe model for concurrent Java programs is available. A first approach to type-safe product lines was proposed in [AKGL10]. For web applications there is a WSDL enhancement in the direction of type systems [LPT06]. In [HHH12] an approach using contracts is introduced for the WSDL composition. The limitation inherent to type systems is that they usually have to be designed to be context sensitive. Security type systems normally associate information such as *secret* or *public* with program parts such as individual instructions or variables. During program execution, however, these parts may process a number of values that may be either *secret* or *public*, depending on the execution context. The granularity of more complex type systems is often too low to reproduce realistic program behavior.

Ultimately, restructuring programming languages towards IT security orientation seems to be the most consequent way. JOE-E [MWC10] presents a first approach for Java.

The challenge for research will be to show what a migration path towards security oriented programming languages could look like and how this path can be pursued consistently [BHLM13], in such a way that it is compatible with the enormous amount of already existing software.

3.2 Challenge: Modeling Risks, Threats and Maturity Levels

Risks become ascertainable, describable and manageable only by modeling the risks, the threats and the maturity levels. There are unfortunately no generally recognized approaches or generally accepted tools for risk, threat and maturity level modeling for the development of secure software products that are not intended for high security areas.

The following list of tools on risk and threat modeling demonstrates that manufacturers are proceeding on differing basic assumptions and origins:

– *TRIKE Threat Modeling Methodology* [SLE05]: TRIKE is a heuristic for threat modeling and can be used for systems and software. TRIKE includes all parties in the risk assessment and a rmation process.

- *CORAS Model-based Method for Security Risk Analysis* [LSS11]: CORAS focusses on risk analysis and can in general be applied to more than just to software (development) alone. It offers a tool based methodology for the model based risk analysis of security relevant systems.
- *Operationally Critical Threat, Asset, and Vulnerability Evaluation for operational risk, not technical risk (OCTAVE)*: OCTAVE only deals with operatiional risks, not technical ones.
- *CCTA Risk Analysis and Management Method (CRAMM)*: The methodology developed by *Central Computing and Telecommunications Agency* (CCTA) is closely tied to using a commercial tool and carries out a threat and vulnerability analysis as well as a risk assessment in order to derive appropriate measures. Since carrying out CRAMM involves significant expenditures, it is considered only as the method of choice for critical systems.
- *AZ/NZS 4360*: AZ/NZS 4360 represents a generic standard for documenting and managing risks. AZ/NZS 4360 includes seven steps: risk context, risk identification, risk analysis, risk evaluation, risk treatment, risk documentation and communication, risk monitoring and reviewing.

The following three frameworks for stating the achieved security levels start from different points as well:

- The *Integrated Measurement and Analysis Framework for Software Security* [AAS10] methodology proposed by the *Carnegie Mellon University's* (CMU) *Software Engineering Institute* (SEI) can be applied to the phases of the software development process.
- The publication [AAS12] gives an overview of the various possibilities to measure the security level.
- *CVSS Common Vulnerability Scoring System* determines vulnerability severity *ex post* as a value between 0 and 10 using a variety of categories.

There is at least one analytical tool to determine the governance maturity when developing secure software: The *Open Software Assurance Maturity Model* (OpenSAMM [Ope13]) is a model for determining the maturity of an organization with regard to secure software processes, thus referring to organizational key figures.

Though manufacturers do offer different tools for risk, threat and maturity modeling, there still remain several aspects that necessitate clarification:

- How can risk, threat and maturity modeling be carried out so that they deliver intersubjectively reproducible results?
- How can it be ensured that objective approaches for risk, threat and maturity modeling will be used more progressively?
- How do this challenge s models interact with the development models of the following challenge? How can risk, threat and maturity models be integrated seamlessly into development models to achieve a secure software lifecycle?

3.3 Challenge: Development Models for Secure Software Lifecycles

Rigorously applied development models raise a software s security level from the very beginning over the complete lifetime [Mic13b]. To realize these frameworks it is essential that they be integrated successively without delays for the development phases and that they mesh in such a way that they appear to the protagonists as if made from one piece, instead of sitting next to each other like a bunch of silos, which was the case until now. Unfortunately no framework features complete and seamlessly integrated assistance systems, and it is not possible to verify or test whether security oriented tools are applied correctly and in a sustainable manner. The following lists frameworks with a high maturity level:

– *Microsoft Security Development Lifecycle (SDL)* [HL06]: According to Microsoft, SDL led to a measurable reduction of the security relevant vulnerabilities [LSP+11]. Each SDL step is supported by tools [Mic13a]. As far as it is known there is no obligation to use a tool for a step. It is not possible to carry out semi-automatic or fully automatic checks to find out whether tools are being used, and only some of the tools are integrated into the development environments.
– *Software Assurance Forum for Excellence in Code (SAFECode):* The SAFECode consortium [SAF07] started with the aim to distribute processes for secure software development industrywide. Examples of SAFECode members are Adobe, CA Technologies, EMC Corporation, Intel Corporation, Microsoft Corp., SAP AG, Siemens AG and Symantec. Recommendations are welcome throughout. It remains open how the detailing, implementation and proof of having carried out the recommendations is done, as well as how the automation of software security by means of tools is approached.

Integrating the following research approaches as tools would close significant vulnerabilities in secure software manufacturing. These approaches present appealing starting points for assistance tools as per the above description:

– *Program Comprehension:* The work at the universities of Stuttgart and Bremen on program comprehension may offer a promising approach in the context of secure software development. Analyses of program behaviors and architectures are to be one part of a secure development process. One possible technical solution may be the Bauhaus project [Bau13]. The security related analyses made possible by this are described by Bunke and Sohr in [BS11].
– *Safety im Softwareentwicklungsprozess:* [RBG12]: SAFE offers a hierarchic programming model that contributes to secure web application enhancement (including the secure personalized code of individual users).

Without doubt, the development models and research approaches mentioned above are useful for raising the software security level from the start. For further development, the following questions need to be answered:

— How can assistance systems for preventing vulnerabilities during the software development process be embedded rigorously and seamlessly into development environments so that existing vulnerabilities may be closed using lifecycle approaches? Once a vulnerability has been detected automatically, tools for the fully automated vulnerability recognition in software manufacturing may prevent known vulnerabilities for the large part, for example by allowing the transmission of a version into the version control system repository only after the vulnerability has been eliminated.

— How can transitions between development cycle phases be formed to ensure that decidedly listed vulnerabilities are not present (anymore)? Ideally such assurances should be fully automatically or alternatively semi-automatically verifiable.

3.4 Challenge: Veri cation and Testing

Ultimately each software has to be tested if it is fulfilling the requirements, in our case if it is *secure*, i. e. does it meet the given security requirements. In view of the software s complexity (and the requirements to be tested!) it is necessary to automatize the verification to the greatest possible extent here as well.

Basically three procedures are available to choose from, all of which have their strengths and weaknesses. *Static Code Analysis* inspects the program code to review all the possible executions of a program. The desired result is that all possible executions meet the (security) requirements; then the program can also be verified as meeting the requirements. Obviously such proof is eminently valuable. Interestingly enough, in IT security a much-cited disadvantage of static analyses transforms into an advantage. Static code analyses abstract from a program s user entries. In other application areas this lack of information about realistic user entries frequently leads to inaccurate analytical results. However, in IT security one has to suspect a malevolent user (the attacker), for whom all possible entries are thus realistic. Static code analyses automatically take such entries into consideration, just like any other.

Unfortunately static analyses have both theoretical barriers and practical problems. The *halting problem* says that there is no general method that can predict any given program s behavior. This is why static code analyses have to work with *approximations*. Depending on the analysis design this may either lead to false alarms or to actually existing problems being overlooked. To construct an analysis in such a way that it recognizes vulnerabilities in any random program at a rate of 100 percent is regrettably not possible.

Another problem in practice is that the static code analysis must know and be able to analyze the complete program text in order to be in a position to make verified statements. Using varied programming languages, distributed or inaccessible programming codes pose an immense challenge for static code analysis. In practice, a technology stack such as web applications (e.g. JavaScript in a browser, PHP-

SQL-C assembler in a server) is not open to current analytical technologies. This is why static code analysis today is usually limited to individual subsystems, the secure functioning of which represents an important basis for the overall system s security. But by now code analyses have reached a high maturity level for such systems. Just recently systems for the static code analyses or more precisely for *Information Flow Control* were introduced and executed successfully on medium sized and large programs, initially the tools JOANA [HS09] and FlowDroid [FAR$^+$13].

The second technique, the *testing*, involves other requirements. Testing necessitates the possibility to execute the program in order to compare the result to the specifications. It is hardly relevant for many testing approaches which programming languages were used for the software testing. Assuming that errors can be detected reliably, testing should not cause false alarms (if the result does not meet the requirements, then there is a problem). The problem of the testing is that only a *finite* quantity of executions may be checked, but the quantity of possible executions is *infinitely* large, which raises a problem in the next new execution, despite best testing.

In practice it comes down to testing as many program behaviors as possible; for this, *test generators* generating the test input data are increasingly employed. Such generators can generate random input (*fuzzing*), but also search very specifically for security vulnerabilities. Modern test generators search very specifically for vulnerabilities that static code analyses have determined as being possible (*DART* / Microsoft), or recombine error causing input (*LangFuzz* / Mozilla) to automatically detect hundreds of security vulnerabilities. However, none of these systems can offer a guarantee for future executions .

The third alternative would be to translocate the test into the actual execution, thus checking the result during every execution, i. e. also during production! This allows for the preempting of faulty results due to constructional conditions. The disadvantages of this *runtime verification* are the increased computation expenditure during runtime and the fact that error situations cannot be recognized and dealt with until execution. At that point in time only little context information is available, which makes it di cult to conduct an expedient error treatment. In real life such runtime tests can be realized at justifiable costs [Bod10], but the static code analysis remains the sole technique that can guarantee the absence of errors ex ante.

Whether static code analysis, testing or runtime tests: Each program analysis needs to know what it is looking for and thus requires a specification of the desired behavior (to search for vulnerabilities within this context) or of the undesired behavior (to search for possibilities to accomplish it). There are a number of program behaviors that tend to lead to an undefined behavior or program termination and therefore are always undesired; for example this allows specifically verifying or testing for buffer over ows. The desired or undesired program behavior has to be exactly specified for example, in the form of a security model describing and restricting users and subsystems precise rights. Just like other specifications such models can become very complex very quickly. This leads to the absurd situation that (provided adequate progress in verification and testing) testing whether a software meets the specification improves, whilst we still do not know whether the specification comprises what is desired or needed.

Given the multitude of challenges it becomes clear that no single approach will su ce on its own. The different program analysis procedures (static code analysis, testing and runtime test) have to work *hand in hand* to play out their respective strengths such as the static code analysis of small subsystems, the interaction of which is to be checked within the respective context using comprehensive tests. The biggest challenge, however, is formulating suitable specifications in such a manner that they are accessible to every programmer. Without specifications there are no errors and thus no correctness, "only" surprises.

Methods for *extracting specifications* from existing systems open up new opportunities currently in the form of axiomatic pre- and post-conditions [ECGN01], finite automatons [DKM^{+}12] or process oriented models [Sch11]. The basic idea is to apply such methods to existing systems and to extract *standardized models* for their behavior (also in the light of security!) from this, in order to check (by means of verification and testing) in what way other systems fulfill these (implicit) standards. With regard to security, the result would no longer be the *infringement* of an explicitly specified security model, but rather the *anomaly* when compared to other (similar) systems. Extracting such detailed specifications is an open research issue; but the experience encoded in billions of programming lines is a prize ready for the taking.

3.5 Challenge: The Sustainably Secure Integration of Cryptographic Primitives and Protocols

Designing complex systems is normally done component-by-component; the enormous complexity of large software projects such as modern multi-user operating systems is not manageable without modularization. Other than in the case of missing functionalities, which can usually be upgraded easily by adding another module, upgrading security features is not quite so effortless. Modularization is often accompanied by an isolated view of individual subsystems, which carries a high inherent security risk. Even if each individual component seems to be "locally secure" it still does not guarantee that the overall system is "globally secure".

This compositional problem exists in two dimensions: In the vertical dimension an attacker will compromise a part of the software stack to gain access to other levels. For example, an attacker breaks into a computer s operating system to manipulate the applications running on that computer. The problem in the horizontal dimension is more subtle but no less significant. Security vulnerabilities in unimportant components may compromise the security of highly critical components (and as such the security of the overall system). For example, the Stuxnet malware was able to use a security vulnerability in the Windows printing system to compromise the whole computer and ultimately to spread over the Busher processing plant.

The way in which local security guarantees can be cancelled out globally by an unsuitable composition in practice has been demonstrated by the attack on the chip-and-PIN procedure [MDAB10]. The chip-and-PIN procedure is a chip card based payment system; the customer places his card in the merchant s pay terminal and authorizes the payment by entering the PIN or by signing a bill. Each available type of authorization can be considered as acceptably secure in itself. The mechanism for choosing between the two modes, however, is implemented in such a way that the card will accept any PIN at signature authorization. In a man in the middle attack a terminal may be tricked into requiring an authorization by PIN, while the card is in signature authorization mode. This means that an attacker may be able to use a stolen card for payment without knowing the valid PIN or without having to forge the signature. He merely has to be able to control the communication between the stolen card and the terminal. For example, this can be achieved during payment at a terminal by using a self-made dummy card that is connected to the stolen card via radio or a hidden cable.

The composition problem becomes especially evident in the TLS key renegotiation attack [RRDO10]. The TLS protocol serves to build and operate an encrypted and authenticated communication link. During an ongoing session it is possible to discard the current key and to negotiate a new key for further communication. In a classic key renegotiation attack an attacker interrupts his victim s TLS secured communication setup and starts his own TLS secured session instead. He initiates a key renegotiation, then proceeds with the victim s communication setup, which had previously been blocked. The emerging connection is effectively encoded and authenticated. On the server side the authentication process is finished. The client, however, is still in the middle of the registration process due to the interruption, and is subsequently still sending out login information. This may lead, for instance, to confidential login information becoming visible as a public text message in a social media portal.

With its universal composability and reactive simulatability models [Can01; BPW07] theoretic cryptography offers an approach for resolving the dilemma: If one of these models is successful in providing formal evidence of security for one of the components, it guarantees the secure use of said components within an arbitrary context. Provable security in the previously mentioned models, however, entails an abundance of disadvantages, which are in con ict with the practical benefits. For one thing, proving security is highly complex in itself, and beyond that it is error-prone. Since all formally conceivable attacks are actually precluded the models are correspondingly strict; often an immense expenditure is necessary to design security provable systems. With regard to e ciency the result falls far short of practically motivated but theoretically insecure adhoc solutions. Typically, a guarantee for the remainder of the requirements can no longer be given, even if only one security requirement has been breached. For all of these reasons these models are *de facto* unsuitable in practice.

A more pragmatic problem-solving approach from software engineering arranges for "contracts" between the individual system components. Each component of a complex system correlates with other components and utilizes or performs services. The security features of the services rendered are regulated in a contract. This ensures at least that no component mistakenly expects specific security features from another component. But how local contracts between components may be derived from global security requirements still remains an open issue. Beyond that, the component contract model makes the reuse of these components in other contexts more cumbersome. It heavily restricts modularity utilization and specifically the issue of securely integrating legacy systems remains unresolved. A prominent example of potential problems arising, when only one individual module is replaced, is represented by the CAN bus for the electronic communication between control devices in automobiles. Originally conceived with the aim to reduce cable harnesses and in consequence the vehicle s weight, security against manipulations from external attackers was not the focus of its development. The bus could be accessed (e. g. for maintenance purposes) only in a tethered way via a plug contact in the vehicle interiors. Critical within this context was the request for wireless maintenance access without the hassle of cumbersome wiring accompanying the universal advent of WLAN and Bluetooth interfaces. Without a suitable security concept and simply by integrating a radio module, a universal communication bus controlling critical components such as the engine control unit or brakes was left open to wireless access from the outside.

In summary, various questions remain open with regard to "secure integration". For one, it is still not adequately resolved to what extent local security requirements at component level may be derived from the global security requirements of an overall system. The same is also true vice versa, where conclusions may be drawn from an individual component s security features onto the absolute maximum possible security guarantees for the overall system. The most practical approach, starting from an abstract overall system, seems to be the development of tools that allow refining the architecture gradually into such detailed modules that simultaneously pave the path for providing proof of the overall system s security based on the individual modules features. Even in a purely intuitive system design this approach is frequently pursued "manually". Currently, though, universal formal tools can give only insu cient support. This, however, does not affect the following two issues: How to identify systematically the essential global security requirements for an overall system, and how to recover reliably the warranted formal security guarantees in the case of legacy systems.

3.6 Challenge: Detecting Intentionally Introduced Vulnerabilities and Provenance Tracking

To increase software security today a certificate is required to assure that a certain software product stems from a trustworthy manufacturer. Without mentioning the problem of forged certificates, which have appeared increasingly over the recent years, there are still several other points of attack inherent to such a procedure: For one, the user would have to know all the suppliers in order to be really able to trust them. On the other hand, a software manufacturer, who in principle is trustworthy and known, may have other interests than the user. Many a time in the past it became public that some software spies on users to a certain degree. For example, mobile apps such as Facebook or Twitter transferred a mobile phone s complete address book onto their servers without the explicit approval of the user, in order to search the address book for known contacts. But insider threats or hackers may infiltrate a program with a code unnoticed as well, thus compromising the program s security.

However, being able to analyze a programs functionality would be better than having to rely on a manufacturer s benignity. Though program analyses are never able to understand a programs full functionality due to the halting problem, certain security statements may be approximated in such a way that a program categorized as secure will definitely be secure, while a program categorized as insecure may really exhibit security vulnerabilities or may not have been su ciently analyzable. The respective techniques are categorized under the key word *language based security.* Especially the information ow control domain provides the option to check programs for vulnerabilities: information ow control reviews whether sensitive data such as an address book may end up in public channels, for example the Internet. This allows for the tracking down of spy programs. Furthermore, information ow control can verify if untrustworthy user entries may affect important program calculations. Unfortunately such injection attacks appear again and again, permitting the attacker to execute arbitrary code capturing complete servers in the Internet and stealing user data, for example credit card numbers.

To execute information control effectively the data origin (*provenance*) has to be known. The provenance will be attached to all computation results depending on this data. Only in this way can it be guaranteed that at the end of a computation it is still known whether it depended on secret entries or whether the computed data may be publicly visible.

The bottom line is to guarantee an end-to-end security which protects sensitive user data over their entire lifecycle. It starts with the encoded storage on a server, data access control, information control during data processing and ends with the encrypted transfer or storage of the results. The goal must be to receive a certificate about a program s provenance and a program has to handle its data in a secure manner as well.

3.7 Challenge: Common Language

Security by Design, i. e. considering security from the very beginning, necessitates that the overall development process is accompanied by documentation recording the security requirements and the already achieved security assurances. These documents will serve as communication over the various development stages and beyond that as the communication between the various disciplines.

Until now, though, it was not ensured that the varying views of the individual disciplines involved were consistent. The individual disciplines differing terminologies, which are not compatible with each other, represent a major hindrance. Colloquial formulations frequently used instead of a common language are not precise enough and lead to misunderstandings. Thus the individual assurances given by the involved disciplines often do not complement each other to form a seamless overall guarantee. This means that really reliable security assurances can be obtained frequently only "locally", for example for individual secure communication connections, the availability of backups or for the correct implementation of a specific functional requirement. If the individual disciplines views are inconsistent, it will not be clear which security guarantee is valid for the overall system.

A perfect example of this problem arising is a bank transfer from 2004 secured by quantum cryptography. Physicists implemented a process which would most definitely prevent an attacker from getting any information about the code. But an attacker was able to modify messages in a very specific way, without having to learn or know the contents. The bank transfer protocol implemented on top of the quantum cryptographic process, however, expected a different security handshake. By wrongly assuming that the secret key transfer would automatically result in a secure bank transfer, the overall protocol became vulnerable and the amounts to be transferred could be modified [BMQS05].

In cryptography it is assumed that implementations are correct. Cryptography only investigates constructional weaknesses that are independent from implementation errors. Program code verification reviews the correctness of an implementation. In most cases these two terms of correctness are not congruent, because the often purely functional specification reviewed in verification does not ensure that the coding material used in the encoding process is good. When bad encoding material has been used an attacker may be able to gain information about the encoded clear text [hei08].

Programming errors may also lead to an illicit information ow. Special code analysis tools (information ow control) may find undesired information ows, but in order to detect such undesired information ows the admissible information ows have to be specified. It is not ensured, however, that such a specification is consistent with the cryptographic specification.

There are already promising approaches in software development that enable the modeling of security aspects during the design stage [Jür02; BDL06; LBD02] and the reviewing of their implementation [JYB08; DPP12]. But these are mostly solutions with a focused area of application. It is a big challenge to find cross-cutting solutions which guarantee that a consistent picture exists during the whole development cycle and across all disciplines involved.

Software verification investigates the relationship between the input of a process versus its output, i. e. the functional features of processes. Security features, however, are non-functional. For example, a successful decoding defines the encoding functionally. The security of an encoding springs from the distribution of tasks, not from their relationship to input. If this vulnerability can be closed the software verification methods may be applied in the IT security area as well.

Normally the security requirements for an overall system are holistically formulated. It is often unclear, what demands these requirements imply for subsystems. On the other hand, it is generally di cult to determine what guarantees may be derived for an overall system from the features of the individual components. It is a challenge to propagate requirements and guarantees equally between the individual development process stages.

In information ow a method has to be found for specifying admissive information ows based on cryptographic requirements and architectural models.

The demand for a common language for the different disciplines raises new issues, for example with regard to the correct degree of abstraction. A high level of detail is necessary for some applications such as cryptographic protocol verification. Such a high level of detail, however, may have a negative impact on other applications due to the overall system s complexity.

It remains open how to extrapolate systematically from abstract colloquial security statements to issues of individual disciplines. A progressively refining methodology in the sense of an attack tree is conceivable.

Due to the increasing juridification of the requirements on IT security the legislator plays an increasingly important role within Security by Design to formulate functional and non-functional requirements on the systems. The distinctive feature is that the legislator generates its own language system in part, the legal terminology, with a compulsory claim to validity. Transforming this legal language into general concepts, while preserving the meaning, is a lawyer s classic type of work. In Security by Design another task is added: Guaranteeing the transformation into the language domains of the various computer science disciplines while preserving the meaning and documenting the transformation processes in a reproducible manner. This task can be accomplished only if all disciplines work together.

The entirety of the approaches pursued by the individual disciplines shall help to evaluate the security of complete systems. It is not known to what extent the disciplines methods of approach are examining all security risks.

4. SECURITY BY DESIGN IN DISTRIBUTED DEVELOPMENT AND INTEGRATION

Present and future software products and IT solutions come from a single developer team only in the rarest of cases as figure 5 shows. For example, within development commissions or by providing open source licenses, other manufacturers may supply software in form of components, libraries and even services that may be combined with proprietary components to larger products. In another aggregate step various products are integrated frequently to create complex IT solutions. For users it is important that the software used by them has the security features they expect, whereas the security requirements and expectations of the different users may vary [FPP12]. Many users with a higher need for security are now looking very closely at what integrators or manufacturers are doing to improve the security of IT solutions or products [Bai12]. But if in turn integrators or manufacturers use the products of other manufacturers, appropriate methods that contribute to the end product s security should be applied along the complete supply chain. Taking the complete supply chain into consideration is important especially because it allows manufacturers to reduce the risk for users with regard to *Advanced Persistent Threats* (APT), in which individualized and specialized attacks are carried out onto selected targets. In the past, such security vulnerabilities, which originated from adequate security processes not being applied during distributed development and integration [Bai12], were frequently the ones used for these types of attack. Even an individual component s security does not represent a su cient provision for the security of the emerging overall product. Vulnerabilities oftentimes occur during integration at the spots where integrated components or products interface. Another problem arises from the integration of open source software, commercial off-the-shelf software (COTS) or legacy code, which is due to the typical market needs of today s software development with regard to time and costs.

To advance the security of integrated solutions and products that have originated in developed distribution suitable procedures and methods are necessary, in which the parts of extremely complex supply chains of software development will be considered. The responsibility to apply such procedures and methods typically lies with the supply chain s last link. But for developing secure software their suppliers have to be included into the security processes as well.

By now the software industry has realized the major significance of security processes along the complete supply chain for secure software and IT solution development. For example, there are activities such as the *Open Group Trusted Technology Forum* [OTT11], which consider software security whilst taking distributed manufacturing processes into account.

Even if today security is increasingly more important to users and manufacturers as a feature and quality characteristic of their IT products and solutions, it is to be noted that manufacturers put in considerably less effort concerning the systematic

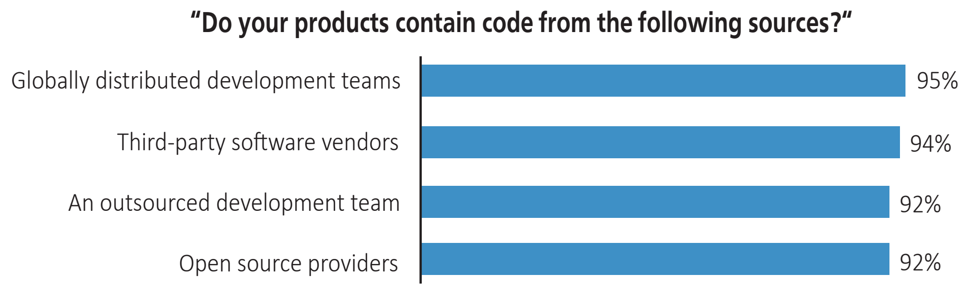


Figure 5: The use of externally developed code (source: [For11b]): The values are based on the survey of 336 IT specialists relevant to software development in their respective company. The companies are located in the USA, Canada, Great Britain, France and Germany.

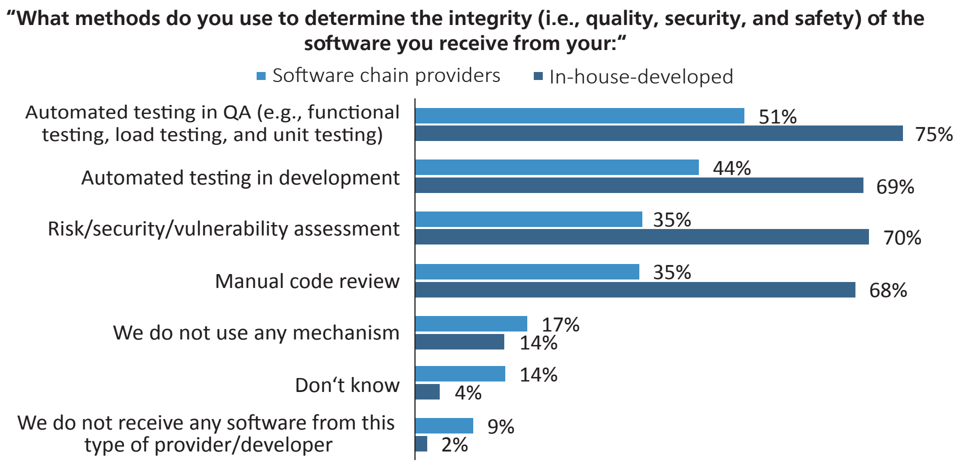


Figure 6: The di erence in quality assurance in internally and externally developed code (source: [For11b]): The basis is the same survey as in gure 5.

and methodically embedded achievement of security in externally developed software components than they do for their own software products. This is evidenced by the findings of a study on the security feature test of externally developed code, displayed in figure 6. The study in figure 6 reviews only the phases that come after the design phase in the software lifecycle. However, it is to be assumed that for the majority of the manufacturers and integrators the prevailing situation regarding the design phase does not differ substantially from the core statement made in figure 6. An important reason for these deficits may be that manufacturers and integrators do not have a uniform standard of procedures and methods with which security processes along the complete supply chain may be realized. Existing security development processes such as Microsoft SDL were not explicitly designed for the distributed development over complex supply chains or for integration [WOUK12].

Secure software product and IT solution development requires uniform and solutions for secure software development processes over the complete supply chain, since components of various manufacturers and components that have been developed following different security processes are embedded today into most of the relevant software products and IT solutions. Approaches referring only to proprietary software development do not su ce to reduce a hacker s chance of success and to improve software security significantly for users [CA11]. Concerning practical application there still is an enormous need for research. The idea of future secure software development is defined by the following vision:

> *The distributed development of secure software and the integration of secure IT solutions will be distinguished by uniform, cross-organizational and security processes along the supply chain, in which security is factored in at the earliest point in time and consistently over the lifecycle.*

Realizing this vision represents an important strategic decision for software manufacturers. On the one hand this decision means for manufacturers that they have to cooperate to enhance security and that they have to depend on their partners to contribute accordingly to the cooperation as well. But cooperation also requires that existing forms of interaction are modified and developed further. On the other hand, such a strategic decision offers software manufacturers the potential to improve their products security accompanied by more favorable development costs. Realizing security processes that incorporate the supply chain represents an important competitive factor for manufacturers. With the increasing significance of software security for the user such security processes represent an important criterion in marketing due to continually increasing compliance guidelines to reduce risks.

For this vision to become reality, a series of challenges has to be mastered, which is described in the following.

4.1 Challenge: Standardizing Security Processes Over The Complete Supply Chain

A coordinated and standardized approach between the various supply chain stakeholders is required to be able to apply security processes in a supply chain management comprising manner. This necessitates standards that have yet to be developed and cover all relevant aspects of distributed development. The following is to be taken into account:

(1) Standardized methods and tools to be used in the security processes
(2) standardized description of the security processes used during component development
(3) standardized description of the security features required from and offered by the components
(4) possibility to review whether security processes are used correctly

Within this context standards will have to cover the entire spectrum of today s distributed development. The spectrum ranges from distributed development in which new software components are being developed within dedicated development assignments, where design and software component development can be guided by the assignments specific requirements, down to the integration of pre-built components such as open source or COTS products. On the one hand, developing such solutions to finished standards represents a substantial challenge that has to be mastered. On the other, such solutions and standards offer a chance for manufacturers and integrators to improve software security, as in doing so they specify strategies and interaction forms, which will not have to be redefined in other individual cases. A unified standard creates a common understanding and congruent ways of thinking among all those involved.

Today s software development world is characterized by very high complexity. Even if the software industry is very globalized and harmonized with regard to specific aspects, the complexity is still determined by things such as differing corporate cultures, peculiarities of the user sector, national and international regulations, different software engineering methods (e. g. agile development), and distinctly different security processes in software development [Bai12]. This complexity is an obstacle that has to be overcome when standardizing the supply chain management dealing with security.

Currently many software industry enterprises have the security process improvement work yet ahead of them. A further reaching approach encompassing the entire supply chain is for most enterprises still far away, even though some software industry representatives and users already understand that software product security measures have to include the software development supply chain. For example, it has already been proposed that a company s risk management needs to take the risk caused by supply chains into account. Works within this context mainly give answers on how to take action against attacks on supply chains, for example in this standard [ISO11] or in [MM08; WLL08; SRM$^+$09]. However, these suggestions for supply chain security are not unique to software products. There is quite a noticeable trend, especially with government organizations as purchasers of software such as end products, components or integration solutions, to scrutinize manufacturer security processes more intensely. For them the existence of suitable security processes is an important criterion when deciding upon certain products or manufacturers [NIS10].

Looking at the security of the software to be used is for users such as enterprises and organizations an essential element in their own security architecture [The11]. When integrating software products from different manufacturers into their own company s infrastructure it is an advantage if integrators can utilize manufacturers statements or assurances about their software s security features. For reasons of effectiveness and e ciency it is important to unify this information ow based on a standard.

More specific suggestions and best practices regarding supply chain security for software, and decision guidance for product evaluations and manufacturers in view of their security processes were given by the *Open Group Technology Forum* in [OTT11]. On the manufacturer side uniform approaches are missing, for example consistent, cross-manufacturer terms or uniform and security processes over the complete supply chain. This makes it di cult to realize these suggestions in practice.

For security processes to function in a supply chain management comprising manner the following questions need to be answered within the scope of the standards:

- How can component security requirements be derived from application security requirements?
- How can the security features for as yet undeveloped and unintegrated components be described simply and e ciently?
- How can the descriptions be fashioned so that they are machine-verifiable while remaining readable for the developers?
- How can component security features and security guarantees be described that were developed for a clearly delineated application and specific environment?
- How can the security features and security guarantees of components be described for which the actual use and environment is not yet known at the time of development and deployment?
- How can it be ensured that all relevant security requirements of components are already included at the design phase?
- How can security processes be standardized across industries and usage?
- How can the cost effectiveness of supply chain management security processes be measured?
- How can product line aspects be included in standards?
- How can it be verified that manufacturers or suppliers comply with the standardized security processes?
- How can manufacturer or supplier violations of the standardized security processes be traced?
- How can manufacturers provide integrators with the relevant product security feature information for a secure implementation?
- How can integrators consolidate the information on security features given to them by the manufacturers and combine them in a beneficial manner?

4.2 Challenge: Governance Framework in Distributed Development and Integration

Governance does not play a fundamental role in software development process restructuring [CA11]. Since software products and integration solutions normally contain software components that were developed by and purchased from third parties, governance frameworks have to regulate how to deal with this. This includes (1) a corporate-wide and transparent regulation of all essential aspects when dealing with other manufacturers software, (2) the responsibilities within this context, and (3) accountability. A governance framework is required for software manufacturers being able to introduce supply chain management security processes corporate-wide. This framework should be harmonized and mandatory within an organization. It has to describe how security processes are to be realized organizationally. The framework has to describe the obligations and responsibilities of all those involved in distinct and transparent regulatory structures.

For various reasons it is absolutely essential to allow the management the control and the responsibility in the governance of an organization:

- Implementing new security processes has a strategic dimension for software manufacturers, independent of it being supply chain management comprising or solely within the corporation. For the manufacturer such security processes have the potential to decrease the financial expenditures over the software s lifecycle while improving the security level. With this in mind such a decision is highly relevant in view of the competition with other manufacturers.
- For certain customer categories existing security procedures are an increasingly significant aspect in their purchasing decision. Especially for manufacturers of software that is used in regulated industries the significance of security processes is particularly important. This involves inasmuch a strategic aspect for software manufacturers that corporate management has to take into account.
- It is well known that security deficiencies in software may have an effect on a manufacturer s stock market listing [TW07; Wri11]. Protecting company values is one of upper management s most essential tasks.
- According to the EU directives EG/2006/48 and EG/2006/49 [EU 06a; EU 06b] that resulted from Basel II, the risks for companies have to be considered when allocating loans. Developing software a icted by security deficiencies may therefore be risky for software manufacturers [Cre11].
- Restructuring software development processes company-wide necessitates a budget that upper management has to be responsible for and provide for.
- Improving application security through security processes requires that software architects and developers apply and realize them across teams and departments. The organization-wide introduction of security processes along a complete supply chain implies that all involved in software development processes will have to

implement the respective guidelines in a concerted manner. This demands upper management level governance.

— Implementing new security processes in software development will change the work developers are accustomed to. In practice, similarly comprehensive processes of change are often marked by resistance aimed at retaining the *status quo*. Against this background, the control and management of introducing new security processes should be established at corporate management level.
— It is only the uppermost management level that can take on the responsibility, at which point in time a standard (see section 4.1) shall be chosen for implementing security processes in the organization.
— The implementation of security processes has to be managed and controlled organization-wide.
— Establishing the framework at the uppermost management level will emphasize the security process restructuring s significance and seriousness in the organization.

The governance frameworks aim at providing companies with a procedure model with which current software development can be improved and conducted by expanding the security processes spanning a complete supply chain. This includes defining new roles and their competencies and responsibilities within the organization. To realize such procedure models, obstacles within the organization need to be identified and removed. Based on the fact that previous approaches and practices in software development will have to be scrutinized, put on trial and modified, opposition and frictional losses are to be expected. Transparency in governance receives a prominent significance against this background, meaning that all involved protagonists will be placed in a position where they can understand the reasons for the further development and the software development process restructuring. This also puts demands on the metrics that are required for managing the further development and restructuring.

To control the introduction of new supply chain management comprising security processes, metrics are needed for identifying progress or problems. At first, suitable metrics need to be developed with which the principal aspects can be measured as effectively, e ciently and objectively as possible. They help management and the executing protagonists in recognizing whether and when the targeted goals have been achieved. Beyond that, the control armamentarium should be su ciently sophisticated to be able to make finely tuned adjustments to individual features. The control and management armamentarium should be repeatedly applicable onto as many departments of an organization as possible.

The governance framework needs to comprise all source relevant in the supply chain of a software manufacturer. In particular, the governance framework must contain recommendations on how suppliers and customers will a rm each other with regards to intertwining security mechanisms, how such a rmations will be made, and how they may be verified.

To achieve the best results possible from one s own software development process restructuring investments it is necessary that third-party suppliers further develop their processes as well and adopt the yet to be developed industry standards (see section 4.1). Including the uppermost management level in such a restructuring will have the favorable effect of being able to in uence other software manufacturers towards adopting the standards.

When developing a governance framework the following questions have to be answered:

— Which rolls are needed in such a governance framework?
— Which processes does the governance framework require?
— Which specific processes does the governance framework require for which type of externally acquired components?
— Which metrics are useful for the governance framework?
— How to increase transparency when implementing the governance framework?
— How to document the governance framework processes?
— How to organize the governance framework in order to restructure software development processes as economically as possible?
— How do security processes with third party suppliers need to be structured at governance level?
— How to verify objectively third party suppliers compliance with the given assurances?

4.3 Challenge: Security Processes for Software Product Lines

The software industry is under massive competitive pressure. Increasing productivity and reducing development time (*time to market*) and development costs are very important for long-term survival. Reusing already developed software components is of great significance within this context.

A specific framework within which the reuse of software components may be planned and organized systematically is inherent in the product lines. Product lines comprise different configurations of a software product that are developed on the basis of a common platform or common kernel for these configurations. These platforms or kernels are then part of all the different product configurations. The different products of a product line develop because platforms or kernels are enhanced at the respective variation points by features. When planning a product line suitable variation points have to be identified at which potential further developments may be added later on. The subjects of such variabilities in product lines are mainly requirements regarding functionality or the compatibility with the environment. Non-functional requirements such as security are normally orthogonal to the development axes and therefore do not have a natural equivalent in systematic product line modeling.

For the manufacturers of more complex software products reuse and product lines play a role, as do distributed development and integration. The *Security by Design* complexity increases if product line and distributed development aspects have to be combined via supply chains.

Various perspectives are relevant when considering product line aspects and supply chains.

(1) For the software component manufacturers as the suppliers within supply chains: The software products developed by a supplier may represent a product within a product line. The development of platforms or kernels as well as product configurations has to be planned and implemented in such a manner by the manufacturer that the requirements of the respective software component customers are being met with regard to their security processes and security features. One di culty in doing so is that the actual requirements of the potential customers may not yet be fully known at the time when the product line is being designed.

(2) For software end product manufacturers and integrators that integrate software components of various manufacturers into their own products: A software end product that has originated from integrating the components of different manufacturers may be a product that has been developed within a product line as well. During product line design security processes and security features have to be considered in such a way that as many relevant security requirements on product configurations can be met as possible. The problem here is also that certain user security requirements are not yet known at the time when the product line is being designed.

When designing product lines and platform security a multitude of security requirements has to be dealt with from the very beginning. These may vary between the different product configurations. First management systems for security requirements in product lines have already been developed [MFMP09; MFMP08a; MFMP08b; MRFMP09] to deal with the systematic handling and administration of these security requirements. Another di culty related to product lines is that threat analyses and actual requirement engineering regarding security can be made for specific use cases only when the platform, on which the product line is based, has been implemented. This makes it possible that specific security requirements may not have been taken into account when the platform security was designed. It may not be ruled out that certain security requirements may not be realizable due to the decisions made regarding the design or the platform. In individual cases it may even be possible that the security design decided upon in a platform may be in direct con ict with the product s security requirements. To avoid security vulnerabilities in products it is therefore always necessary to check the application security requirements against the platform security features. This is also why it is important when dealing with product lines that the security processes typical in software development are adapted to a product line s particularities. Supporting the implementation of these processes with appropriate tools should be very helpful (see chapter 3).

A product line design has to find a good balance between future configurations security requirements that may need to be met and e ciency and cost effectiveness issues, among other things. If too strong an emphasis is placed onto potential security requirements there is a danger of over-engineering, leading to the product line s development costs becoming too high, which in turn will prevent to benefit from the savings potential inherent to the product line approach.

Product lines are characterized by enabling a great number of possible software products, if multifold variation points are available. This means that for *Security by Design* many different configurations have to be considered and analyzed. Results [BRT^{+}13] dealing with the security of such product configurations that may be achieved by varying pre-processor options already exist. This is a first important step towards *Security by Design* in product lines. Further research has to follow, that is not limited to the pre-processor option variation and that takes the problems of distributed software development into account as well.

In order to take product lines security processes and security features in distributed development into account research has to answer the following questions:

— How to fashion the security processes over the complete supply chain in software development while taking product lines into account?

— How to design product lines in such a way that as many relevant security requirements as possible can be met with a reasonable effort?

— How to deal with the security requirements of future product configurations not yet known at the time when the security is being designed?

— How to identify special product configurations and their specific security requirements in product line design?

— How to fashion security analysis tools in such a way that they exploit the common features of different products e ciently, while identifying vulnerability classes at the same time that arise from variability?

— How to identify effectively and e ciently inconsistencies with a product configuration s later issued security requirements in a product line platform s security design?

— How can an integrator transfer a product line platform s security requirements into security requirements for the components manufactured by third-party suppliers?

— Which documentation formats are needed for supply chain management comprising product line security processes?

4.4 Challenge: Security when Integrating Large Systems

In modern enterprises software systems are used in many business work ows. They support business processes, making them more effective, productive and accurate. Today s companies are not competitive anymore without the appropriate software support. Software systems have a decisive advantage when differing business processes can share common data and when the same data and functions can be accessed within these business processes. This allows integrating different applications, which is also called *Enterprise Application Integration* (EAI). With EAI it is possible to react swiftly and exibly to new requirements by enhancing or modifying the existing software systems. Furthermore, EAI offers enterprises the foundation for integrating business processes beyond corporate boundaries. The potential EAI offers to companies has been known for a while [Gle05]. This is true for companies from both the manufacturing sector and the service sector as well [Xu11]. All persons responsible for organizing IT infrastructures in enterprises have to deal with the questions and issues of EAI. These questions and problems arise from the ever increasing degree of integration when compared to earlier information systems that were limited to selected functions and partial integration.

Through the high degree of integration, EAI typically results in very large and complex systems that are customized specifically to the respective user s requirements, thus integrating business processes, the respective phases and configurations of which meet an organization s particular requirements. On the technological level differing components such as systems, applications, interfaces (for example user interfaces) or data, which may be very heterogenoeus, are implemented via EAI to form complex processes. The integration tends to be di cult and costly, because the components were developed using different methods for different systems, because they do not support common interfaces or because they are based on different data models. Integrating components and subsystems that in themselves are very heterogeneous exacts from developers and integrators a high manual effort, which rarely follows unified, systematic and structured procedures due to the heterogeneity. According to estimates integrations today require more than 30% of the overall investment users spend on their IT infrastructure [ROB11]. Benefits and effects of utilization for an organization result from the functionality, which is why functionality always takes priority with EAI.

Today EAI is used intensely for *Enterprise Resource Planning Systems* (ERP) that cover important business processes for enterprises [NTD12]. Beyond ERP systems, software for *Customer Relationship Mananagement* (CRM), *Supply Chain Management* (SCM) or cross-company business processes (B2B) is being used if required. This software is implemented via EAI as well. In many companies universal ERP products are being used as the basis for large software systems. These products were developed for a wide range of users and have functionalities such as integrated data storage, standard applications (e. g. for personnel matters, sales, accounting, production) and general business process implementations. Beyond that there are industry and branch related ERP system configurations [WXH09]. For the typical recurring issues regarding business processes, all of these ERP systems offer solutions in the form of *Best Practices* or established standards, and allow adequate specializations for the respective enterprise (*Customization*). But in many cases the range of functions offered by universal ERP software products does not fully cover the users requirements and wishes, so that additional software products are also being integrated [SS05].

Providing services within service-oriented architectures also offers the possibility to use functionality made available over the Internet through other provided services [WL11]. Recommendations for the integration of services even go so far as to implement services from different providers dynamically and adaptively [MRFU11]. The differing needs, the dynamic, the exibility and the different technological implementations of the user-specific integration of additional components yields complex information systems that differ immensely in their integrated conditions between the different users, even if the same ERP products are used.

With the wide deployment of EAI the risks for the users with regard to vulnerability exploitation increases substantially as well. Components or subsystems of the systems created by EAI facilitate access to critical information. For the companies, the large systems originating from the integration are comparable to a digital treasure room, because they basically comprise all the information from the relevant business processes. The resulting systems are highly complex, which makes it di -cult to assess all the implications for security. It cannot be ruled out that attackers may gain access to data via components or subsystems, running counter to a company s security regulations. Points of origin for attacks may occur especially where the integrated components interface. There exists for neither the initial integration nor for the overall life cycle explicit systematic approaches and methods in the sense of *Security by Design*. In practice IT security issues do not play an essential role in integration [KT09]. Studies show that security vulnerabilities tend to develop during integration because of very simple and avoidable errors [Kal12].

Existing systems for integration refer to the architectural level and describe how to implement components into the overall environment and how they interact. Other systems describe coordination models and the use of tools for integrating data and complex processes [ROB11; Gle05; HN08]. Available research, however, does not include comprehensive security processes for integration. When security is being considered then it is usually limited to considering security standards such as the Web Service Security standards [OAS12] as important technical elements for the secure composition of net based services. Further propositions offer promising approaches for improving security based on the compositional requirements and assurances descriptions of the components to be integrated, for example compositional security contracts as described in [KT09], but they have not yet been devised adequately or transferred into practice.

When integrating large systems the general conditions for *Security by Design* depend strongly on the integration models. The spectrum of what is possible is very large: It ranges from integrating locally available software components, which the applying enterprise may have helped to develop, to the integration of locally installed third-party software, and to integrating software components in the form of services offered by third parties for access over the Internet, for example as cloud services. When using services from third-party providers, the risk for the user increases, when data is stored with service providers whose platform is used by many other customers as well, customers that may possibly be attackers, potentially using vulnerabilities to attack another user s data. Depending on the integration model the options for *Security by Design* differ strongly. When developing large systems approaches and methods should be applied to further improve and maintain the security of the resulting systems over the complete life cycle, independent of the integration model used. This also needs to factor in agility, exibility and economical practicability for future enhancements and adaptations in integration. To accomplish this, suitable techniques have to be developed for the different integration models. Among other things the following challenges will have to be mastered within this context:

- How to realize component security requirements and express them in a comprehensible and applicable manner?
- How to realize assurances regarding security and express them in a comprehensible and applicable manner?
- To what extent do security requirements and assurances need to be treated so that they may be applied in an economical scope for future large system changes and modifications?
- How to derive and realize decisions systematically regarding the architecture and design of the to be developed connective technology during integration from the security requirements of the to be implemented overall processes and their respective components?
- When integrating functionality as a service, how do the processes have to be established to facilitate that the security implications for the remaining components can be determined from the technological modifications, preferably before the technological modifications are implemented?
- How can existing procedures for planning and coordinating integrative work be complemented in order to facilitate *Security by Design*?
- How to take security aspects into consideration for the dynamic integration of services?
- How to adapt existing service descriptions to the dynamic integration in order not to select services that impinge against the security requirements of the remaining system?

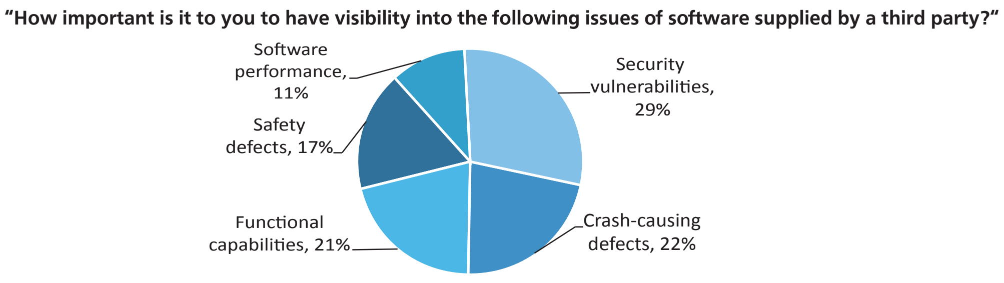


Figure 7: The signi cance of security with implemented software components that were developed by other manufacturers (source: [For11b]): This is based on the same survey as the ones in gures 5 and 6.

4.5 Challenge: Assurance through Security Processes

Software security is a criterion that is becoming more and more important for a user s purchasing decision. This is particularly true for users with big market power, for example public authorities and other government institutions, and for users from specific industries that have to apply stricter rules, the observance of which the organization and management are liable for.

Within this context a user is always interested in the security of the whole end product, even if the end product contains components from different manufacturers and suppliers. From a user s perspective, it is always the manufacturer of the end overall product who responsible for its features, since he is the one who selected the third party components. Accordingly manufacturers and integrators also need to take security issues into account when deciding on suppliers or the software components to be implemented. In such decisions, questions about security are very important for software end product integrators and manufacturers; the results from surveys within the software industry [For11b] demonstrate this as is shown in figure 7.

Software component customers need statements from their suppliers that allow them to assess a component s security level. Such statements should provide an appropriate level of detail and have binding character. In practice, though, it is di cult to make statements about the absolute security level of software products, especially when software products are generated by composing parts from various manufacturers. Information about security processes carried out at the time of the manufacture are another alternative to assure manufacturers, integrators or users that security aspects were taken into account during software manufacturing. Manufacturers should use such assurances to make statements about the extent, and the accuracy and diligence that were applied to specific systems to ensure security. Such assurances are especially helpful if they can be verified in audits as unequivocally as possible and, if breached, the transgressor has to fear negative consequences when infringements against said assurances have been proven.

Thus assurances based on security processes provide an indirect statement on software security. The assurance that certain security processes are adhered to during manufacturing suggests a higher security level. Such assurances based on security processes, for example certifying security measures during manufacturing, stand in contrast to the certification of products, for example on the basis of *Common Criteria*. During certification a direct statement is being made on software product security at different assurance levels, with Common Criteria these are the Evaluation Assurance Levels. Even if a direct statement on software product security, for example using *Common Criteria*, seems to be more appropriate initially than indirect assurances based on the manufacturing processes, practical experience presents some arguments that speak for the indirect approach versus the direct approach. According to [Jac06] certifications by *Common Criteria* are too cumbersome, time-consuming and very expensive. Certification by *Common Criteria* is therefore only used in niche areas, especially in cases where there are particularly high security requirements, for example, because of restrictions due to regulation. According to BSI in [BSI12] certification by *Common Criteria* is used for software products such as operating systems, data bases, firewalls, PC security products, VPN products, e-mail servers and signature application components. For application software, manufacturers avoid the effort and expense caused by *Common Criteria*. A number of fundamental problems play a role that result from *Common Criteria* and are in con ict with the requirements of the software manufacturing companies. Software manufacturers tend to have serious time pressure to market their products. This time pressure is in direct contrast to the considerable delays caused by *Common Criteria*. Additionally, once software products are on the market they are usually developed further continuously in small steps, delivered to the users within updates. However, direct certifications such as with *Common Criteria* imply that the assurance is not valid anymore if a new product update or software version has been released. Manufacturers have to pass the time-consuming and expensive certification process again for each update and each new software version. Another important feature of *Common Criteria* contradicting the application software manufacturers requirements is caused by *Common Criteria* not supporting exible compositions, such as they result from composing a software product with the components from various manufacturers. Assurances or statements regarding security features are especially in practice a very important requirement for such products, because a very large part of real software products combines components from different manufacturers.

The software product world today is thus split into two parts regarding assurances about IT security features: For special products with high security requirements certificates are available that issue statements directly on a product s security features. For typical application software without specific requirements such statements do not exist, so there are no assurances for end product manufacturers, integrators and users to which they might refer.

This situation could be improved by assurances regarding the security processes used during manufacturing. It would be possible that such an assurance would still be valid when a product is further developed with the respective security processes.

It would also be possible, provided adequate conditions, that assurances referring to manufacturing processes may remain valid when composing more complex products. This could generate a benefit in exactly those cases in which other certification methods such as *Common Criteria* may not be able to meet the practical requirements.

Even if assurances based on the applied security processes do not allow to generate direct statements about security features, indirect approaches maybe very valuable, as they can give assurances and statements about lots of products for which today there are no usable statements on security. Beyond that studies showed, as described in section 2.4, that the systematic application of security processes improved software product security distinctly.

In view of the distributed development processes, software component manufacturers can provide end product manufacturers with assurances based on the security processes. This requires developing a suitable framework which stipulates for the various products sensible and precisely describable steps (for example methods for requirements engineering, design methods, security tests) and specific criteria for the respective procedure (e. g. considering certain relevant vulnerability compilations such as OWASP `http://www.owasp.org` within the context of the web applications that have to be taken into account in tests; frequency of tests; using accredited tools that support developers in their programming by preventing certain programming errors). Within this framework it is furthermore important that crucial parts of the security process can be audited. The auditability of assurance compliance allows to accord assurances the necessary commitment. Suppliers may otherwise simply claim that they carry out certain processes without actually doing so.

To achieve such a commitment it should su ce if the effort to circumvent auditable security process is essentially the same as for implementing them. On the other hand, the supplier needs to be sure that the solution for auditable security processes is secure against violations that may be construed against the assurances once all assurances have been implemented correctly.

For the framework on assurances and auditability the following issues have to be mastered among others:

– How to determine the relevant assurances for different software components and different areas of use in an e cient way?

– How to verify that the relevant assurances were identified for the area of use?

– How to precisely phrase assurances?

– How to ensure that suppliers and integrators speak the same language in their assurances?

– What repercussions do assurances have on the design of security processes? (From various conceivable security process variants the one should be prefered that is the most economical one to fulfill assurances)

- How to make assurances in such a way that compliance or violations are verifiable?
- How to recognize without a doubt assurance violations?
- How to trace violations to the originator with absolute certainty?
- How to verify assurance violations and compliance e ciently?
- How to secure auditable assurances against deceit?
- How to reconcile assurances and the auditability of assurances with other solutions for security processes spanning the complete supply chain?
- How to verify a supplier s security processes over the entire lifecycle even after the software components have been delivered?
- How to achieve assurances by tool support?
- How can assurances based on the applied security processes be renewed when methodology and tools are further developed?

5. SECURITY BY DESIGN FOR LEGACY SOFTWARE

In their book about legacy software [SPL03] Robert C. Seacord, Daniel Plakosh and Grace A. Lewis use the term *legacy crisis* to demonstrate powerfully the growing challenges regarding legacy software. Whether it is economical to continue using already existing software, or if the required functionality may need to be programmed completely anew is a decision that has many facets. For example documentation completeness and quality, platform independence, programming language independence, and the comparison between the targeted and the current status of the achieved security level. A minimum security standard is a necessary prerequisite for software reuse or continued use.

This chapter s aim is focused on the security revision of legacy software:

> *A required stipulation for the reuse or continued use of legacy software is its IT security revision; legacy software may be used only when an adequately high security level exists for its area of use. For a decision-making basis plausible statements have to be issued about the present IT security level. For reuse or continued use the software must be introduced into the security lifecycle. For the continued use of existing software it will be much easier to introduce a higher security standard.*

5.1 Challenge: Statements about the Security of Legacy Software

Given the increasing demand to integrate legacy software, statements about the security of legacy software are urgently needed (compare also [SPL03], chapter 4 and 5). Whether the security level of legacy software may indeed be determined remains open: Years later a single undetected programming error may turn out to be security relevant. This does not only mean that software is basically run insecurely, but it is even argued that it is impossible in principle to determine the security of software [Bel06].

Even if it is not possible to ascertain the security level of software intersubjectively and down to the last detail, it must at least be possible to make a plausible judgment in order to be able to perform a risk assessment. Only by doing so, can it be decided whether legacy software may be used further on or in a new context at a specific minimum security level.

Current approaches to assess security levels are going in different directions and there is no measurement process that is accepted as state of the art in technology and research.

For example, at source code level these include three approaches that differ methodologically:

- BogoSec (*source code security quality metrics*) [KS06] uses instrumented test tools for source code analysis, which are applied in combination and from which an aggregated security level is computed.
- According to [CCZ08] the source code structure analysis generates statements on the basis of continuous compliance with programming principles.
- Michael A. Howard, on the other hand, proposes a completely different method: a comparative *code review* [How06]. Depending on the overlap rate, he assesses the number of yet undiscovered vulnerabilities in an experimental setting with two development teams

If the software is not yet available as source code then assessing the security level is obviously an even bigger challenge [PC10; Sav10]. For example, it could be checked if the accordingly adapted experimental setting [How06] could be a candidate for a measurement process here as well. Can *Software Penetration Testing* [ASM05] be modified in such a way that it can be used for legacy software as well and that statements about the security level are possible? Can relevant assessment tools [Boo09] be adapted in such a manner that they permit statements about the achieved security level?

In view of the previously mentioned although very promising and diverse first ideas, the research is still at the very beginning with regard to the measurability of the legacy software s security level. There is a substantial need for research.

Several crucial questions remain open, for example:

- Which measurement processes represent plausible candidates for statements on the security level of legacy software?
- Are the statements about the IT security level easy to communicate and do they provide real decision guidance concerning the continued use or reuse of legacy software with regard to security?
- How big is the measuring effort (time, resources)?
- When is it practical to carry out the measurement?
- Is the measurement robust, valid and intersubjectively repeatable?

5.2 Challenge: Transfer Legacy Software into the Security Lifecycle

Legacy software that is to be reused or continued to be used and is not yet in part of the security lifecycle must be introduced into it. The full integration of legacy software into a process where it is systematically tracked, and where vulnerabilities are monitored and reviewed (e. g. the systematic *Common Weakness Enumeration* (CWE) [MIT13]) is particularly important.

An issue that is not to be underestimated is the question of how to identify entry points for legacy software in the respective lifecycle so that security considerations and measures become feasible for the overall software in an integrated manner, after the introductory phase. A first approach for such entry points has been given by CLASP s *Legacy Roadmap* [Gra06].

With *IBM Internet Security Systems Product Lifecycle Policy* [IBM06] IBM has presented a body of rules for their proprietary software s security aspects, which has the advantage that legacy security aspects are already taken into consideration during software manufacturing.

To introduce legacy software systematically into a security lifecycle, the following questions need to be answered independently from manufacturers:

— How to prepare software already during its development for a later secure reuse or continued use?
— How can manufacturers devise policies for old software reuse and continued use that allow the links in the supply chain to integrate the old software more easily?

5.3 Challenge: Increase the Security of Legacy Software

Software that was manufactured with only little or no security issues in mind and that will remain in use often needs to be upgraded to a (higher) security level. Various proposals have been made to increase the security level of legacy software. Which of these are effective and e cient remains to be seen. A systematic analysis and comparison is urgently needed. along with possible enhancements.

If the source code exists, most options are of course available for hardening, especially when the source code is very well documented. The spectrum ranges from complex analyses and subsequent security hardening by experts (*Source Code Review*), to a fully automatic hardening through source code replacements. From an economical point of view the latter one is especially interesting. Examples for measures on different levels:

— Incremental type security: Increasing the type security of existing programs is a first sensible step. *Gradual Typing* starts with the insecure program and adds type systems incrementally [ST07].
— Programming language related hardening: The following shows two examples for source code related measures, one for C, one for Java. CCURED [NCH$^+$05] enhances the storage and type security of C source code by rewriting code in security critical program parts. To harden Java source codes [MLD08] the other focuses on an aspect oriented approach via *Hardening Patterns.*
— In legacy software, enhancements to enforce security policies may be supported by specific program analysis tools [GJJ06]. For example by automatic code rewriting [Ham06] applied to *Managed Code* of the .NET-framework. Another example is the hardening of security policies by *Web Services* [MOA11] via automatic BPEL aspect generation (*Business Process Execution Language*).
— Runtime Monitoring can encapsule legacy components and thus ensure that they fulfill specific policies [Bod12].

The usual techniques for integrating black box legacy software securely [SM99] such as system call analysis and prevention (e. g. [LRB+05] and [RHJS05]), wrapper, sandboxes, firewalls and instrumentation (for example by monitors) are currently receiving a promising amendment by the *SecondWrite* tool [OAK+11], which overwrites code for *Black Box Executables* at security critical positions on the lower system level with secure code.

It is definitely possible to increase the security level of legacy software, as has been demonstrated there is a number of measures available for this. The following questions need clarifying:

- How to determine with very little effort if a hardening is worthwile, or if for example a complete reprogramming would be more expedient?
- How to categorize legacy software, so that suitable hardening measures can be assigned to the resulting categories? For example, categories may be programming languages, the software technology used, age of the software, but also the maturity level and the completeness of the documentation.

6. THE FUTURE WITH SECURITY BY DESIGN

Do we really want to keep reading news about new security vulnerabilities and attacks on a daily basis? Do we want to continue using software, in the manufacturing of which security hardly played a role, even though computer and software are becoming more and more relevant for many areas of our everyday life? How long do we still want to put up with this catch-up game between hackers and manufacturers, the victims of which tend to be the users? Changing the *status quo* is up to the manufacturers, the users, society, and the political establishment.

A promising way out of this situation is *Security by Design.* History shows that production processes could already be modified successfully elsewhere : The chemical industry does not discharge their untreated sewage into rivers anymore, and by now all vehicles cause less environmental pollution because of the reduced emission of pollutants. Similar changes should also be possible for the production processes of secure software.

Security by Design is characterized by offering benefits for all those involved: software becomes more secure, risks decrease, the costs for manufacturing and maintenance are reduced, and the manufacturing companies gain more competitiveness. Security can become an important added value in the software manufacturing process.

The future will be about investigating the deciding issues concerning *Security by Design* and developing utilizable solutions. This involves industry, research, and the political establishment. Large companies need to take on the leadership role, because medium-sized software manufacturers are not capable of reorganising their production processes under their own control.

With these ideals this trends and strategy report set a direction in which *Security by Design* can and must develop. Beyond that, the report describes challenges that have to be dealt with and problems that need to be resolved. These visions and challenges will shape the cyber security research agenda in the coming years.

This requires the software industry, the research community, and the political establishment to close ranks in order to produce utilizable results that are target and application oriented, and to transfer these into practical software manufacturing.

7. APPENDIX: BIBLIOGRAPHY

REFERENCES

[AAS10] Alberts, C.; Allen, J. ; Stoddard, R.: *Integrated measurement and analysis framework for software security.* White Paper, SEI CERT,`http://www.cert.org/archive/pdf/10tn025.pdf`, 2010

[AAS12] Allen, J.; Alberts, C. ; Stoddard, R.: *Deriving Software Security Measures from Information Security Standards of Practice.* White Paper, SEI CERT,`http://www.sei.cmu.edu/library/assets/whitepapers/derivingsecuritymeasures.pdf`, 2012

[Abe10] Aberdeen Group: *Security and the Software Development Lifecycle: Secure at the Source.* `http://www.microsoft.com/en-us/download/confirmation.aspx?id=6968`, 2010

[Ado13] Adobe Systems Incorporated: *Secure Product Lifecycle.* `http://www.adobe.com/de/security/splc/`. Version: 2013

[AKGL10] Apel, Sven; Kästner, Christian; Grö linger, Armin ; Lengauer, Christian: Type safety for feature-oriented product lines. In: *Automated Software Engineering* 17 (2010), September, Nr. 3, S. 251 300

[ASM05] Arkin, Brad; Stender, Scott ; McGraw, Gary: Software penetration testing. In: *IEEE Security & Privacy* 3 (2005), Nr. 1, S. 84 87

[Bai12] Baize, Eric: Developing Secure Products in the Age of Advanced Persistent Threats. In: *IEEE Security & Privacy* 10 (2012), Nr. 3, S. 88 92

[Bau13] Bauhaus-Projekt: *Software-Architektur, Software-Reengineering und Programmverstehen.* `http://www.iste.uni-stuttgart.de/ps/projekt-bauhaus.html`. Version: 2013

[BBMM10] Bruch, Marcel; Bodden, Eric; Monperrus, Martin ; Mezini, Mira: IDE 2.0: collective intelligence in software development. In: *Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER '10)*, 2010

[BDL06] Basin, David; Doser, Jürgen ; Lodderstedt, Torsten: Model driven security: From UML models to access control infrastructures. In: *ACM Trans. Softw. Eng. Methodol.* 15 (2006), Januar, Nr. 1, S. 39 91

[Bel06] Bellovin, S.M.: On the Brittleness of Software and the Infeasibility of Security Metrics. In: *IEEE Security & Privacy* 4 (2006), Nr. 4, S. 96

[BHLM13] Bodden, Eric; Hermann, Ben; Lerch, Johnannes ; Mezini, Mira: *to appear: Reducing Human Factors in Software Security Architectures.* `http://www.future-security2013.de/`. Version: 2013

[BKA11] BKA (Bundeskriminalamt): *Wirtschaftskriminalität — Bundeslagebild 2010.* `http://www.bka.de/nn_193360/DE/Publikationen/JahresberichteUndLagebilder/Wirtschaftskriminalitaet/wirtschaftskriminalitaet__node.html?__nnn=true`, 2011

[BKA12] BKA (Bundeskriminalamt): *Cybercrime — Bundeslagebild 2011.* `http://www.bka.de/DE/Publikationen/JahresberichteUndLagebilder/Cybercrime/cybercrime__node.html?__nnn=true`, 2012

[BMQS05] Beth, Thomas; Müller-Quade, Jörn ; Steinwandt, Rainer: Cryptanalysis of a practical quantum key distribution with polarization-entangled photons. In: *Quantum Information & Computation* 5 (2005), Nr. 3, S. 181 186

[BMW12a] BMWi (Bundesministerium für Wirtschaft und Technologie): *Monitoring-Report Digitale Wirtschaft 2012 — MehrWert für Deutschland.* `http://www.bmwi.de/BMWi/Redaktion/PDF/Publikationen/it-gipfel-2012-monitoring-report-digitale-wirtschaft-2012-langfassung,property=pdf,bereich=bmwi2012,sprache=de,rwb=true.pdf`, 2012

[BMW12b] BMWi (Bundesministerium für Wirtschaft und Technologie): *Nationaler IT-Gipfel 2012: digitalisieren_vernetzen_gründen (Essener Erklärung).* `http://www.bmwi.de/BMWi/Redaktion/PDF/Publikationen/it-gipfel-2012-essener-erklaerung,property=pdf,bereich=bmwi2012,sprache=de,rwb=true.pdf`, 2012

[Bod10] Bodden, Eric: E cient Hybrid Typestate Analysis by Determining Continuation-Equivalent States. In: *ICSE '10: International Conference on Software Engineering*, 2010, 5 14

[Bod12] Bodden, Eric: *Project RUNSECURE.* `http://www.ec-spride.tu-darmstadt.de/csf/sse/projects_sse/emmy_noether/emmy_noether.en.jsp`, 2012

[Boo09] Booz Allen Hamilton: *Software Security Assessment Tools Review*, März 2009

[BPW07] Backes, Michael; Pfitzmann, Birgit ; Waidner, Michael: The reactive simulatability (RSIM) framework for asynchronous systems. In: *Inf. Comput.* 205 (2007), Nr. 12, S. 1685 1720

[BRT$^+$13] Bodden, Eric; Ribeiro, Márcio; Tolêdo, Társis; Brabrand, Claus; Borba, Paulo ; Mezini, Mira: SPLLIFT Statically Analyzing Software Product Lines in Minutes Instead of Years. In: *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2013

[BS11] Bunke, Michaela; Sohr, Karsten: An architecture-centric approach to detecting security patterns in software. In: *Engineering Secure Software and Systems.* Springer, 2011, S. 156 166

[BSI06] BSI (Bundesamt für Sicherheit in der Informationstechnik): *M 2.378 System-Entwicklung.* `https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02378.html`. Version: 2006

[BSI12] BSI (Bundesamt für Sicherheit in der Informationstechnik): *Zertifizierte IT-Sicherheit.* `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Broschueren/ZertIT/zertifizierte-IT.pdf?__blob=publicationFile`, Oktober 2012

[BSI13] BSI (Bundesamt für Sicherheit in der Informationstechnik): *Lageberichte des Bundesamts für Sicherheit in der Informationstechnik (BSI).* `https://www.bsi.bund.de/DE/Publikationen/Lageberichte/lageberichte_node.html`, Januar 2013

[CA11] Chess, B.; Arkin, B.: Software Security in Practice. In: *IEEE Security & Privacy* 9 (2011), March-April, Nr. 2, S. 89 92

[Can01] Canetti, Ran: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: *Proceedings of FOCS 2001*, 2001, S. 136 145. Revised version online available at `http://eprint.iacr.org/2000/067`

[CCZ08] Chowdhury, Istehad; Chan, Brian ; Zulkernine, Mohammad: Security metrics for source code structures. In: *Proceedings of the fourth international workshop on Software engineering for secure systems (SESS '08)*, 2008

[Chr11] Christley, Steve: *CWE//SANS Top 25 Most Dangerous Software Errors.* `http://cwe.mitre.org/top25/`, 2011

[Cov13] Coverity: *Annual Coverity Scan Report.* `http://softwareintegrity.coverity.com/register-for-the-coverity-2012-scan-report.html`. Version: 2013

[Cre11] Creative Intellect Consulting: *Failure to invest in secure software delivery puts businesses at risk.* businesswire, `http://www.businesswire.com/news/home/20110223006536/en/Failure-Invest-Secure-Software-Delivery-Puts-Businesses`, Februar 2011

[DKM$^+$12] Dallmeier, Valentin; Knopp, Nikolai; Mallon, Christoph; Fraser, Gordon; Hack, Sebastian ; Zeller, Andreas: Automatically Generating Test Cases for Specification Mining. In: *IEEE Trans. Softw. Eng.* 38 (2012), März, Nr. 2, S. 243 257

[DMN12] DMN (Deutsche Mittelstands Nachrichten): *Angriff auf Online-Banking: Hacker stehlen 36 Millionen Euro von Privatkunden.* `http://www.deutsche-mittelstands-nachrichten.de/2012/12/48673/`, 2012

[DPP12] Denney, Ewen; Pai, Ganesh ; Pohl, Josef: Heterogeneous Aviation Safety Cases: Integrating the Formal and the Non-formal. In: *Proceedings of the 2012 IEEE 17th International Conference on Engineering of Complex Computer Systems (ICECCS '12)*, IEEE Computer Society, 2012, S. 199 208

[ECGN01] Ernst, Michael D.; Cockrell, Jake; Griswold, William G. ; Notkin, David: Dynamically discovering likely program invariants to support program evolution. In: *IEEE Transactions on Software Engineering* 27 (2001), Februar, Nr. 2, S. 99 123

[EU 06a] EU (Europäische Union): *RICHTLINIE 2006/48/EG DES EUROPÄISCHEN PARLAMENTS UND DES RATES vom 14. Juni 2006 über die Aufnahme und Ausübung der Tätigkeit der Kreditinstitute.* Amtsblatt der Europäischen Union L 177/1, 2006

[EU 06b] EU (Europäische Union): *RICHTLINIE 2006/49/EG DES EUROPÄISCHEN PARLAMENTS UND DES RATES vom 14. Juni 2006 über die* *angemessene Eigenkapitalausstattung von Wertpapierfirmen und Kreditinstituten.* Amtsblatt der Europäischen Union L 177/201, 2006

[FAR+13] Fritz, Christian; Arzt, Steven; Rasthofer, Siegfried; Bodden, Eric; Bartel, Alexandre; Klein, Jacques; le Traon, Yves; Octeau, Damien ; McDaniel, Patrick: *Highly Precise Taint Analysis for Android Applications. Technical Report.* `http://www.bodden.de/pubs/TUD-CS-2013-0113.pdf`, Mai 2013

[FIB13] Frost & Sullivan; (ISC)^2 ; Booz Allen Hamilton: *The 2013 (ISC)^2 Global Information Security Workforce Study.* `https://www.isc2.org/workforcestudy/Default.aspx`, 2013

[For11a] Forrester Consulting: *State of Application Security.* `http://www.microsoft.com/en-us/download/confirmation.aspx?id=2629`, 2011

[For11b] Forrester Research: *Software Integrity Risk Report — The Critical Link Between Business Risk And Development Risk.* `http://www.coverity.com/library/pdf/Software_Integrity_Risk_Report.pdf`, April 2011

[FPP12] Fichtinger, Barbara; Paulisch, Frances ; Panholzer, Peter: Driving Secure Software Development Experience in a Diverse Product Environment. In: *IEEE Security & Privacy* 10 (2012), Nr. 2, S. 97 101

[GJJ06] Ganapathy, V.; Jaeger, T. ; Jha, S.: Retrofitting legacy code for authorization policy enforcement. In: *2006 IEEE Symposium on Security and Privacy*, 2006

[Gle05] Gleghorn, Rodney: Enterprise Application Integration: A Manager s Perspective. In: *IT Professional* 7 (2005), November, Nr. 6, S. 17 23

[Gra06] Graham, Dan: *The CLASP Application Security Process.* `https://buildsecurityin.us-cert.gov/bsi/100/version/1/part/4/data/CLASP_ApplicationSecurityProcess.pdf?branch=main&language=default`, 2006

[Ham06] Hamlen, Kevin: *Security policy enforcement by automated program-rewriting.* Ithaca, NY, USA, Diss., 2006

[hei08] heise Online: *Schwache Krypto-Schlüssel unter Debian, Ubuntu und Co.* `http://www.heise.de/security/meldung/Schwache-Krypto-Schluessel-unter-Debian-Ubuntu-und-Co-207332.html`. Version: Mai 2008

[hei11] heise Security: *Angriff auf Playstation Network: Persönliche Daten von Millionen Kunden gestohlen.* `http://www.heise.de/security/meldung/Angriff-auf-Playstation-Network-Persoenliche-Daten-von-Millionen-Kunden-gestohlen-1233136.html`, April 2011

[hei12a] heise Security: *Chinesische Hacker gingen bei Nortel ein und aus.* `http://www.heise.de/security/meldung/Chinesische-Hacker-gingen-bei-Nortel-ein-und-aus-1433741.html`. Version: 2012

[hei12b] heise Security: *Immer mehr EU-Bürger haben Angst vor Cyber-Kriminalität.* `http://www.heise.de/security/meldung/Immer-mehr-EU-Buerger-haben-Angst-vor-Cyber-Kriminalitaet-1635864.html`, Juli 2012

[hei13] heise Security: *Schwerwiegende Sicherheitslücke bei Amazon.* `http://www.heise.de/security/meldung/Schwerwiegende-Sicherheitsluecke-bei-Amazon-1786722.html`, Januar 2013

[HHH12] Hollunder, B.; Herrmann, M. ; Hülzenbecher, A.: Design by Contract for Web Services: Architecture, Guidelines, and Mappings. In: *International Journal On Advances in Software* 5 (2012), Nr. 1 and 2, S. 53 64

[HL06] Howard, Michael; Lipner, Steve: *The Security Development Lifecycle.* Redmond, WA, USA : Microsoft Press, 2006

[HN08] Haase, Thomas; Nagl, Manfred: Service-Oriented Architectures and Application Integration. In: *Collaborative and Distributed Chemical Engineering. From Understanding to Substantial Design Process Support - Results of the IMPROVE Project* Bd. 4970. Springer, 2008, S. 727 740

[How06] Howard, Michael: A Process for Performing Security Code Reviews. In: *IEEE Security & Privacy* 4 (2006), Juli, Nr. 4, S. 74 79

[HS09] Hammer, Christian; Snelting, Gregor: Flow-Sensitive, Context-Sensitive, and Object-sensitive Information Flow Control Based on Program Dependence Graphs. In: *International Journal of Information Security* 8 (2009), Dezember, Nr. 6, S. 399 422

[IBM06] IBM: *IBM Internet Security Systems Product Lifecycle Policy.* `http://www-935.ibm.com/services/us/iss/pdf/support_product_lifecycle_policy.pdf`. Version: June 2006

[IBM12] IBM: *IBM X-Force 2012 Mid-year Trend and Risk Report.* `http://www-935.ibm.com/services/us/iss/xforce/trendreports/`, September 2012

[ISO11] ISO (International Standardization Organisation): *Security management systems for the supply chain – Development of resilience in the supply chain – Requirements with guidance for use.* `http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=56087`, 2011

[Jac06] Jackson, Joab: *Symantec: Common Criteria is bad for you.* `http://gcn.com/Articles/2007/05/04/Symantec-Common-Criteria-is-bad-for-you.aspx?p=1`, 2006

[Jür02] Jürjens, Jan: UMLsec: Extending UML for Secure Systems Development. In: *Proceedings of the 5th International Conference on The Unified Modeling Language (UML '02)*, 2002

[JYB08] Jürjens, Jan; Yu, Yijun ; Bauer, Andreas: Tools for traceable security verification. In: *Proceedings of the 2008 international conference on Visions of Computer Science: BCS International Academic Conference (VoCS'08)*, 2008, 367 378

[Kal12] Kallus, Michael: *5 Sicherheitsschwachstellen in SAPSystemen.* CIO Magazin `http://www.cio.de/2889344`, August 2012

[KS06] Kirkland, Dustin; Salem, Loulwa: *BogoSec: Source Code Security Quality Calculator.* `http://sourceforge.net/projects/bogosec/`, März 2006

[KT09] Khan, Khaled M.; Tan, Calvin: SecCom: A Prototype for Integrating Security-Aware Components. In: *Information Systems: Modeling, Development, and Integration, Third International United Information System Conference, UNISCON 2009, Sydney, Australia, April 21-24, 2009. Proceedings* Bd. 20, Springer, 2009 (Lecture Notes in Business Information Processing), S. 393 403

[LBD02] Lodderstedt, Torsten; Basin, David A. ; Doser, Jürgen: SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: *Proceedings of the 5th International Conference on The Unified Modeling Language (UML '02)*, 2002

[Loc12] Lochbihler, Andreas: *A Machine-Checked, Type-Safe Model of Java Concurrency : Language, Virtual Machine, Memory Model, and Verified Compiler*, Karlsruher Institut für Technologie, Fakultät für Informatik, Diss., Juli 2012

[LPT06] Lapadula, A.; Pugliese, R. ; Tiezzi, F.: A WSDL-based type system for WS-BPEL. In: *Coordination Models and Languages* Springer, 2006, S. 145 163

[LRB$^+$05] Linn, C. M.; Rajagopalan, M.; Baker, S.; Collberg, C.; Deinsty, S. K. ; Hartman, J. H.: Protecting against unexpected system calls. In: *In Proceedings of the 14th USENIX Security Symposium*, 2005, S. 239 254

[LSP$^+$11] Ladd, David; Simorjay, Frank; Pulikkathara, Georgeo; Jones, Jeff; Miller, Matt; Lipner, Steve ; Rains, Tim: *The SDL Progress Report.* `http://www.microsoft.com/en-us/download/details.aspx?id=14107`, 2011

[LSS11] Lund, Mass S.; Solhaug, Bjørnar ; Stølen, Ketil: *Model-Driven Risk Analysis - The CORAS Approach.* Springer, 2011

[McG06] McGraw, Gary: *Building Secure Software.* Addison Wesley Professional Computing, 2006

[MDAB10] Murdoch, Steven J.; Drimer, Saar; Anderson, Ross J. ; Bond, Mike: Chip and PIN is Broken. In: *IEEE Symposium on Security and Privacy (S&P 2010)*, 2010

[MFMP08a] Mellado, D.; Fernández-Medina, E. ; Piattini, M.: Security Requirements Variability for Software Product Lines. In: *Third International Conference on Availability, Reliability and Security(ARES '08)*, 2008, S. 1413 1420

[MFMP08b] Mellado, Daniel; Fernández-Medina, Eduardo ; Piattini, Mario: Towards security requirements management for software product lines: A security domain requirements engineering process. In: *Computer Standards & Interfaces* 30 (2008), Nr. 6, S. 361 371

[MFMP09] Mellado, Daniel; Fernández-Medina, Eduardo ; Piattini, Mario: Security Requirements Management in Software Product Line Engineering. In: *e-Business and Telecommunications, International Conference, ICETE 2008, Porto, Portugal, July 26-29, 2008, Revised Selected Papers*, 2009

[Mic10] Microsoft: *Secure Development Lifecycle — Simplified Implementation of the Microsoft SDL.* `http://download.microsoft.com/download/F/7/D/F7D6B14F-0149-4FE8-A00F-0B9858404D85/Simplified%20Implementation%20of%20the%20SDL.doc`, 2010

[Mic13a] Microsoft: *Microsoft Security Development Lifecycle Tools.* `http://www.microsoft.com/security/sdl/adopt/tools.aspx`, Januar 2013

[Mic13b] Microsoft: *SDL Helps Build More Secure Software.* `http://www.microsoft.com/security/sdl/learn/measurable.aspx`, 2013

[MIT13] MITRE: *Common Weakness Enumeration.* `http://sourceforge.net/projects/bogosec/`, Februar 2013

[MLD08] Mourad, Azzam; Laverdière, Marc-André ; Debbabi, Mourad: An aspect-oriented approach for the systematic security hardening of code. In: *Computers and Security* 27 (2008), Nr. 3-4, S. 101 114

[MM08] Manuj, Ila; Mentzer, John T.: Global Supply Chain Risk Management. In: *Journal of Business Logistics* 29 (2008), Nr. 1, S. 133 155

[MOA11] Mourad, A.; Otrok, H. ; Ayoubi, S.: Toward Systematic Integration of Security Policies into Web Services. In: *2011 European Intelligence and Security Informatics Conference (EISIC)*, 2011, S. 220 223

[MRFMP09] Mellado, Daniel; Rodriguez, J.; Fernández-Medina, E. ; Piattini, M.: Automated Support for Security Requirements Engineering in Software Product Line Domain Engineering. In: *International Conference on Availability, Reliability and Security,(ARES '09)*, 2009, S. 224 231

[MRFU11] Mukhija, Arun; Rosenblum, David S.; Foster, Howard ; Uchitel, Sebastián: Runtime Support for Dynamic and Adaptive Service Composition. In: *Rigorous Software Engineering for Service-Oriented Systems - Results of the SENSORIA Project on Software Engineering for Service-Oriented Computing* Bd. 6582. Springer, 2011, S. 585 603

[MWC10] Mettler, Adrian; Wagner, David ; Close, Tyler: *Joe-E: A Security-Oriented Subset of Java.* `http://joe-e.org/`, 2010

[NCH$^+$05] Necula, George C.; Condit, Jeremy; Harren, Matthew; McPeak, Scott ; Weimer, Westley: CCured: type-safe retrofitting of legacy software. In: *ACM Trans. Program. Lang. Syst.* 27 (2005), Mai, Nr. 3, S. 477 526

[NIS10] NIST (National Institute for Standards): *Guide for Applying the Risk Management Framework to Federal Information Systems — A Security Life Cycle Approach.* NIST Special Publication 800-37 Rev. 1, `http://csrc.nist.gov/publications/nistpubs/800-37-rev1/sp800-37-rev1-final.pdf`, Februar 2010

[NTD12] Nazemi, Eslam; Tarokh, Mohammad J. ; Djavanshir, G.Reza: ERP: A Literature Survey. In: *The International Journal of Advanced Manufacturing Technology* 61 (2012), S. 999 1018

[Nü12] Nüsse, Andrea: *Revolution per Kurznachricht.* `http://www.zeit.de/politik/ausland/2012-01/aegypten-revolution-jahrestag`, Januar 2012

[OAK$^+$11] O Sullivan, Pádraig; Anand, Kapil; Kotha, Aparna; Smithson, Matthew; Barua, Rajeev ; Keromytis, AngelosD: Retrofitting Security in COTS Software with Binary Rewriting. In: *Future Challenges in Security and Privacy for Academia and Industry* Bd. 354. Springer Berlin Heidelberg, 2011

[OAS12] OASIS Web Services Security Maintenance TC: *Web Services Security v1.1.1.* OASIS Standards, https://www.oasis-open.org/standards#wssv1.1.1, Mai 2012

[ON98] Oheimb, David von; Nipkow, Tobias: Machine-checking the Java Specification: Proving Type-Safety. In: *Formal Syntax and Semantics of JAVA*, Springer, 1998, S. 119 156

[Ope13] OpenSAMM: *Open Software Assurance Maturity Model.* http://www.opensamm.org/. Version: 2013

[OTT11] OTTF (The Open Group Trusted Technology Forum): *Open Trusted Technology Provider Framework (O-TTPF) — Industry Best Practices for Manufacturing Technology Products that Facilitate Customer Technology Acquisition Risk Management Practices and Options for Promoting Industry Adoption.* http://www.opengroup.org/ottf, Februar 2011

[PC10] P eeger, S.L.; Cunningham, R.K.: Why Measuring Security Is Hard. In: *IEEE Security & Privacy,* 8 (2010), Nr. 4, S. 46 54

[RBG12] Reischuk, Raphael M.; Backes, Michael ; Gehrke, Johannes: SAFE Extensibility for Data-Driven Web Applications. In: *WWW'12: Proceedings of the 21st International Conference on World Wide Web.* Lyon, France, 2012

[RGWS08] Reichenbach, Gerold; Göbel, Ralf; Wolff, Hartfrid ; Stokar von Neuforn, Silke: *Risiken und Herausforderungen für die öffentliche Sicherheit in Deutschland — Grünbuch des Zukunftsforums Öffentliche Sicherheit — Szenarien und Leitfragen.* http://www.zukunftsforum-oeffentliche-sicherheit.de/downloads/Gruenbuch_Zukunftsforum.pdf, 2008

[RHJS05] Rajagopalan, Mohan; Hiltunen, Matti; Jim, Trevor ; Schlichting, Richard: Authenticated System Calls. In: *In Proc. IEEE International Conference on Dependable Systems and Networks (DSN2005)*, 2005

[ROB11] Rodrigues, Nuno; Oliveira, Nuno ; Barbosa, Luís S.: The role of coordination analysis in software integration projects. In: *On the Move to Meaningful Internet Systems (OTM 2011)* Bd. LNCS 7046, Springer-Verlag, October 2011, S. 83 92

[RRDO10] Rescorla, E.; Ray, M.; Dispensa, S. ; Oskov, N.: *Transport Layer Security (TLS) Renegotiation Indication Extension.* RFC 5746 (Proposed Standard). http://www.ietf.org/rfc/rfc5746.txt. Version: Februar 2010

[SAF07] SAFECode (Software Assurance Forum for Excellence in Code): *SAFECode.* http://www.safecode.org/index.php, 2007

[Sav10] Savola, Reijo: On the Feasibility of Utilizing Security Metrics in Software-Intensive Systems. In: *IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1* (2010)

[Sch11] Schur, Matthias: Experimental specification mining for enterprise applications. In: *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering,(ESEC/FSE '11)*, 2011

[SLE05] Saitta, P.; Larcom, B. ; Eddington, M.: Trike v. 1 methodology document. (2005). http://www.octotrike.org/papers/Trike_v1_Methodology_Document-draft.pdf

[SM99] Souder, T.; Mancoridis, S.: A tool for securely integrating legacy systems into a distributed environment. In: *Proceedings. Sixth Working Conference on Reverse Engineering*, 1999, S. 47 55

[Spi12] Spiegel Online: *Industriespionage bei Nortel — Chinesische Hacker sollen Tech-Konzern ausgeplündert haben.* http://www.spiegel.de/netzwelt/web/industriespionage-bei-nortel-chinesischehacker-sollen-tech-konzern-ausgepluendert-haben-a-815102.html, 2012

[Spi13] Spiegel Online: *Monatelanger Angriff — Chinesische Hacker spähten „New York Times" aus.* http://www.spiegel.de/netzwelt/netzpolitik/new-york-times-monatelange-angriffechinesischer-hacker-a-880654.html, 2013

[SPL03] Seacord, R.C.; Plakosh, D. ; Lewis, G.A.: *Modernizing legacy systems: software technologies, engineering processes, and business practices.* Addison-Wesley Professional, 2003

[SRM+09] Simpson, Stacy; Reddy, Dan; Minnis, Brad; Fagan, Chris; McGuire, Cheri; Nicholas, Paul; Baldini, Diego; Uusilehto, Janne; Bitz, Gunter; Karabulut, Yuecel ; Phillips, Gary: *Software Supply Chain Integrity Framework — Defining Risks and Responsibilities for Securing Software in the Global Supply Chain.* SAFECode Publication, http://www.safecode.org/publications/SAFECode_Supply_Chain0709.pdf, 2009

[SS05] Schelp, Joachim; Schwinn, Alexander: Extending the business engineering framework for application integration purposes. In: *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC)*, 2005, S. 1333 1337

[ST07] Siek, J.; Taha, W.: Gradual typing for objects. In: *ECOOP 2007–Object-Oriented Programming* (2007), S. 2 27

[Tas02] Tassey, Gregory: *The economic impacts of inadequate infrastructure for software testing.* NIST (National Institute of Standards and Technology), Planning Report 02-3, 2002

[The11] The Open Group TOGAF-SABSA Integration Working Group: *TOGAF and SABSA Integration — How SABSA and TOGAF complement each other to create better architectures.* White Paper, Reference W117, https://www2.opengroup.org/ogsys/catalog/w117, Oktober 2011

[TW07] Telang, Rahul; Wattal, Sunil: Impact of Software Vulnerability Announcements on the Market Value of Software Vendors An Empirical Investigation. In: *Workshop on the Economics of Information Security (WEIS'07)*, 2007

[VK11] Vorgang, Blair R.; Karry, Alec: *Addressing Software Security in the Federal Acquisition Process.* Cigital White Paper, https://www.cigital.com, 2011

[WAZ12] WAZ: *Hacker nutzen immer öfter Sicherheitslücken bei Behörden.* http://www.derwesten.de/wirtschaft/digital/hacker-nutzen-immer-

`oefter-sicherheitsluecken-bei-behoerdenid6408800.html`, Februar 2012

[WL11] Wu, Zhuang; Li, Yan: Research on enterprise application integration based on Web. In: *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, 2011, S. 2221 2224

[WLL08] Williams, Zachary; Lueg, Jason E. ; LeMay, Stephen A.: Supply chain security: an overview and research agenda. In: *International Journal of Logistics Management* 19 (2008), August, Nr. 2, S. 254 282

[WOUK12] Wataguchi, Yoshiro; Okubo, Takao; Unno, Yukie ; Kanaya, Nobuyuki: Cooperative Secure Integration Process for Secure System Development. In: *15th International Conference on Network-Based Information Systems (NBiS 2012)*, 2012, S. 782 786

[Wri11] Wright, Craig S.: Software, Vendors and Reputation: An Analysis of the Dilemma in Creating Secure Software. In: *Trusted Systems - Second International Conference, INTRUST 2010, Revised Selected Papers* Bd. LNCS 6802, Springer, 2011, S. 346 360

[WXH09] Wu, Shi L.; Xu, Lida ; He, Wu: Industry-oriented enterprise resource planning. In: *Enterprise Information Systems* 3 (2009), Nr. 4, S. 409 424

[Xu11] Xu, Li D.: Enterprise Systems: State-of-the-Art and Future Trends. In: *IEEE Transactions on Industrial Informatics* 7 (2011), Nr. 4, S. 630 640

[Zel07] Zeller, Andreas: The Future of Programming Environments: Integration, Synergy, and Assistance. In: *2007 Future of Software Engineering(FOSE '07)*, 2007

ACKNOWLEDGEMENTS

This trends and strategy report has been supported by the German Federal Ministry of Education and Research (BMBF) in the context of the competence centers for Cybersecurity

- European Center for Security and Privacy by Design (EC SPRIDE, `http://www.ec-spride.de`),
- Center for IT-Security, Privacy and Accountability (CISPA, `http://www.cispa-security.de`) und
- Competence Center for Applied Security Technology (KASTEL, `http://www.kastel.kit.edu`)

The authors express their deepest gratitude to the German Federal Ministry of Education and Research for this support.

Furthermore they thank Mr. Thomas Caspers from the Federal O ce for Information Security (BSI) for his valuable suggestions. Also they thank Anne Grauenhorst (CASED und EC SPRIDE), Alex Wöhnl (EC SPRIDE), Viktoriia Kunetska (EC SPRIDE) and Sarah Ahmed (CASED) for their support. For their support with the English translation of this report the authors thank Thomas Eldridge (CASED), Matthew Elverson (CASED) and Ben Hermann (CASED and EC SPRIDE).

Competence Centers for Cybersecurity

In order to take on the large challenges in Cybersecurity, the German Federal Ministry of Education and Research (BMBF) chose three competence centers: CISPA, EC SPRIDE and KASTEL. They combine the excellence of the best universities and research facilities in the field of Cybersecurity research both thematically and organizationally. The competence centers are funded by the German Federal Ministry of Education and Research (BMBF) since 2011. Despite the centers all working on different areas of the the field they are strongly cooperating.

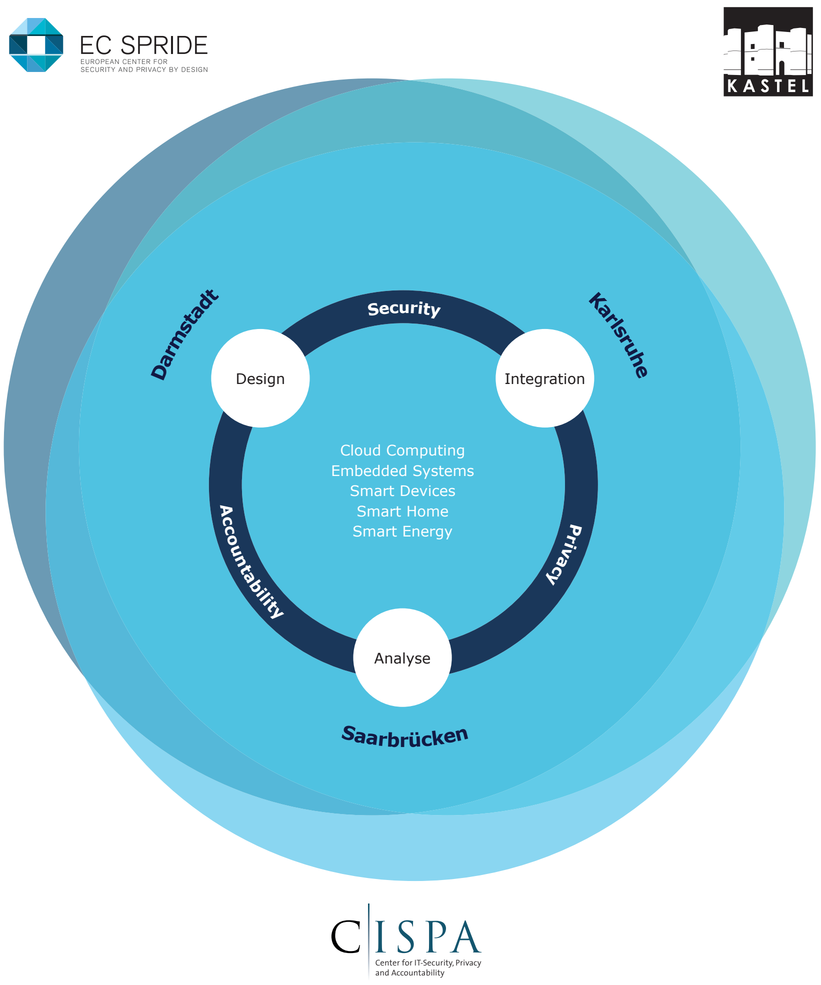


Figure 8: The competence centers for cybersecurity

CISPA (Saarbrücken)

The Center for IT-Security, Privacy and Accountability (CISPA) aims to develop a holistic approach to solve the central IT Security issues of the digital society. It combines basic research for the analysis of existing and the discovery of new approaches with their systematic advancement into a universal toolbox of deployable security technologies for complex systems. CISPA s core topics are reliable security, accountability, and privacy.

EC SPRIDE (Darmstadt)

The European Center for Security and Privacy by Design (EC SPRIDE) researches into how IT developers can optimally secure software and IT systems from the very beginning i.e. by Design and throughout the entire lifecycle. The researchers in the *Engineering*, *Building Blocks* and *Blueprint* research areas compile the basic knowledge and create new development and testing methods for ensuring optimal software security. In doing so, they also take the latest technical and social developments into consideration as relevant practical parameters.

KASTEL (Karlsruhe)

The Competence Center for Applied Security Technology (KASTEL) researches into how secure applications can be developed in one integrated process. Innovative solutions will be demonstrated by three prototypes in areas that are highly relevant for our society: Cloud Computing, Smart Energy and privacy protecting camera surveillance. To achieve this, researchers from computer science will closely cooperate with Economic Sciences and Law. The center aims to turn away from isolated partial solutions and instead develop an integrated approach that combines competencies and methods from different disciplines.

ISBN 978-3-8396-0768-8