Riskpool – A Security Risk Management Methodology

Martin Ring*, Paul Duplys*, Sven Köhler†
*Robert Bosch GmbH
email:{martin.ring, paul.duplys}@de.bosch.com
†ITK Engineering GmbH
email:{sven.koehler}@itk-engineering.de

***Abstract*—Risk management is widely defined as a process during product development. As an example, the International Organization for Standardization (ISO) 31000 family of standards defines risk management as *coordinated activities to direct and control an organization with regard to risk*. While necessary, process-related aspects cover only one part of a risk management system since processes usually specify *that* something must be done, but not how to do it. In this paper, we propose a new methodology for implementing risk management in commercial software engineering, over the complete product lifetime. We illustrate our method by showing how it can be applied to address cyber security risks. We argue that our method has significant advantages over classical risk management techniques especially in domains like cyber security where new regulations and laws are being introduced.**

Keywords-security; risk management; cyber security.

I. Introduction

Traditionally, risk management methodologies for product development were designed with the (implicit) assumption that products do not change after being deployed in the field. As a result, these methodologies have limited means to address maintenance and support – including keeping the product's software up-to-date – over the entire lifetime of a product as mandated by new regulations like [1]–[6].

Conceptually, risk management methodologies from other industries, e. g., insurance, can be applied to product development. In practice, however, there is no industry consensus how to estimate the initial risk of all active products and, in particular, the *cost* associated with this risk. In addition, insurance companies typically adjust their risk assessment and the corresponding insurance fee on a yearly basis. For cyber-physical products, on the other hand, such adjustments are impractical because substantial price variations from year to year are unacceptable from the customers' perspective. Consequently, the initial risk assessment needs to be more precise and the risk management methodology has to take into account that the cost-risk function must remain valid over a multi-year period.

In this paper, we describe a new methodology that allows to derive a more reliable initial risk assessment and a sustainable cost function. In addition, we give hints for the recurring assessment. In Section II, we present the state of the art in risk management and risk assessment, and describe the challenges faced by companies offering products that include software or are themselves software-based. In Section III, we introduce our approach to generating and maintaining a holistic risk management system. Additionally, we provide an example and discuss the results in detail. Section IV concludes the paper by summarizing our approach and offering an outlook for future developments.

II. State of the Art

In the following sections, we want to introduce the background information necessary and to further motivate the presented approach.

A. Risk Management

Risk management became more and more the risk management of everything, but the focus shifted in the mid 1990's from managing first order risks (risks directly stemming from the developed product, e. g., radiation in mobile phones or food quality) to second order risks (the public perception of the company when first order risks manifest) [7]. With this paper, we want to lay the focus on first order risk management. For most domains, the definition and probability for these risks can be calculated, e. g., for safety mechanisms in the automotive domain, there are a priori known probabilities for failures, these have to be minimized below an acceptable level and these ratings or levels do not change over lifetime of the product and will not degrade. On the contrary, cyber security risks can change over the life time of a product and need constant effort to keep a connected product safe and secure, as new vulnerabilities are discovered and hacker capabilities increase, e. g., with increased computing power. So, not only is it necessary to reevaluate the associated risk of such a product, but laws and regulations [1]–[3] mandate that known vulnerabilities need to be addressed, at least in part, by updates to the product software over the *customer expected lifetime*. Discussion regarding this definition is ongoing, but lifetimes for consumer products are expected to be between two and ten years, depending mainly on their price.

B. Common Vulnerabilities and Exposures (CVE) + Code

To get a ballpark figure of how many vulnerabilities over the course of a product's lifetime might need to be fixed, we take a look at a diverse set of software projects and their assigned vulnerabilities. The overview is summarized in Table I, while Figure 1 gives an overview over typical sizes of code bases for different products. The table was created with CVEs over a time span of 10 years (2012-2022). Depending on the company planning to apply this methodology, the time span can be adjusted, but a general trend is visible in constantly increasing code bases and, with that, at least a similar increase

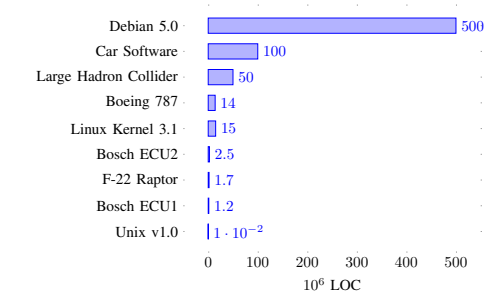

Figure 1. Lines of code of different software projects [10].

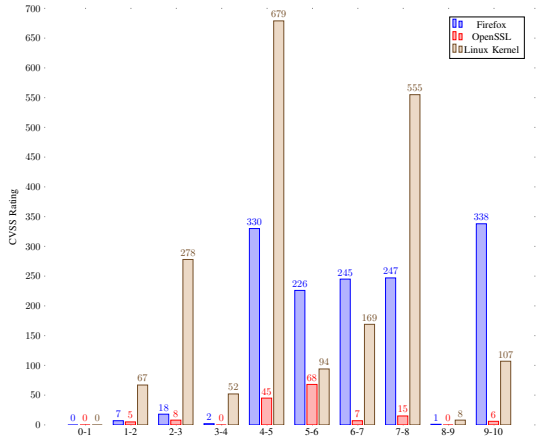

Figure 2. CVSS Ratings for three software projects 2011-2021 [11].

in vulnerabilities is expected. Other sources put the defect rate of software as industry average in the range of 1 to 25 defects per 1,000 Lines Of Code (LOC) [8]. Not all these defects will result in vulnerabilities, but they all potentially can lead to a vulnerability. When we take a cautious estimation of 100,000,000 (LOC) in a modern car [9] and take some of the best (lowest number of CVEs per 1,000 LOC) software as basis for our assumption (firefox) we come to a total of at least 5,766 exploitable vulnerabilities over the supported lifetime (ten years) of a modern car. These vulnerabilities have no impact rating at the moment assigned to them. The distribution of Common Vulnerability Scoring System (CVSS) ratings is plotted for the sample projects in Figure 2. The ratings are from 2011 to 2021, because 2022 still had a lot of unrated CVEs. As can be seen, no clear distribution is visible, thus we can learn from this overview, that there is a typical bandwidth of vulnerabilities per LOC, and this value depends also on the period under review, but there is no generally applicable distribution of vulnerability distribution regarding their risks (CVSS ratings). To come to a reasonable risk estimation, we need further information from the project, the basis for which will be presented in the next section.

C. Threat Analysis and Risk Assessment

A Threat Analysis and Risk Assessment (TARA) is standard for almost all modern software projects, but not limited to those. A TARA strives to formally enumerate all possible

TABLE I. OVERVIEW OF BIG AND DIVERSE SOFTWARE PROJECTS, THEIR NUMBER OF CODE LINES, REPORTED AND CVE ASSIGNED NUMBER OF VULNERABILITIES [11], [12].

Software	Vulnerabilities per 1,000 LOC	LOC (2023)	∑ CVEs 2012-2022
Google Chrome	0.08	25,600,000	2,154
Firefox	0.06	25,300,000	1,459
Linux Kernel	0.07	33,600,000	2,230
OpenSSL	0.11	1,540,000	163
Python	0,06	1,320,000	77
PHP	0.23	1,510,000	349

TABLE II. RISK MATRIX EXAMPLE [3].

		Attack Feasibility Rating			
		Very Low	Low	Medium	High
Impact Rating	Severe	2	3	4	5
	Major	1	2	3	4
	Moderate	1	2	2	3
	Negligible	1	1	1	1

threats to a product (e. g., based on attack trees [3]). There are multiple norms, standards and publications that target this specific topic [3], [13], [14]. Only most recently the ISO 21434 provided a way to rate vulnerabilities with safety implication, while, e. g., the common criteria approach solely focuses on the attack potential, and CVSS ratings only partially address the impact ratings, especially in cyber-physical-systems, as highlighted before [15], [16]. The CVSS version 4.0 now includes safety aspects but does not include them in the final scoring. The ISO 21434 proposes a rather simple matrix, where impact and feasibility are rated from negligible to severe, and very low to high, respectively, see Table II - resulting in a single dimension risk value. This risk value (points) will be one of the values that defines the inequation of the risk pool we introduce below.

Risk in general can be *avoided*, *reduced - impact/likelihood*, *shared* and *retained*. Risk avoidance and reduction is possible, e. g., with the introduction of further mitigations, that possibly increase the LOC count, but reduce at the same time the sum of residual risks of the TARA. Risk sharing is not in the focus of this publication, although it is possible to distribute the fallout of a defect. Risk retention is the acceptance of certain risks, as the costs of fixing them would outweigh the possible costs of defects in the field [17].

III. A RISK POOL AS RISK MANAGEMENT METHODOLOGY

The general idea of this methodology consists of the risks associated with all products that are not at the end of their lifetime on one hand and a risk pool, representing the available capacity to fix defects in a product over its lifetime on the other hand, described in more detail with Equation (1)

$$\sum_i (\text{Project}_i \cdot \text{TARA Residual Risks}_i \cdot \text{Weight}_i) \leq \sum \text{Developers} \cdot \text{Fixing Capability} \cdot \text{Capacity} \quad (1)$$

Where:

- i: i is the number of all projects in the period under review (i. e., the lifetime of the product associated with the project). Each project with a different risk, lifetime or changed code base has to be counted individually.
- *TARA Residual Risks*: As described before, the vulnerabilities of different projects follow no common distribution, but to get to a good first estimation we take the residual risks as basis for our inequation to determine the risk pool.
- *Weight*: These residual risks will be weighted with an additional value based on five properties (LOC, code age, innovation level, update capability & known defects [17]). The weight can be set between 0.02 and 0.1. This weight allows a scaling according to the aforementioned properties, and thus a weight regarding exploitability, i. e., a new and unreleased product that might be of high interest to adversaries should be weighted with 0.1.
- *Developers*: The number of software developers in the considered company
- *Fixing Capability*: this capability will depend on the risk evaluation methodology and the product.
- *Capacity*: How much of the work force or how much of their time should or can be assigned to fixing defects that resulted in vulnerabilities.

A. Example

For demonstration purposes, we will look into the fictitious company ExCom; this company has two products, ECU 1 – an Automotive Safety Integrity Level (ASIL) D and ECU 2 – an ASIL B ECU. We will have a look at these two products including their risk rating and we will look at how a risk pool would look like for this company.

1) ECU 1 - ASIL D ECU: This product is of the highest safety category for automotive applications, but it is a rather simple ECU with a smaller code base. The innovation level of this ECU is low, the code base is older and some deviations are already in production and in the field, so the code base is well tested and proven. Furthermore, the connectivity is low, as it has only the capability for low range external communication. The risk was determined according to ISO 21434 and as depicted in Table II. The sum in this project according to this method is a weighted TARA residual risk value of 29 points, with a weight of 0.02, based on the aforementioned terms.

2) ECU 2 - ASIL B ECU: This product has a medium safety category, it is a completely new product, with a high innovation level. The connectivity is low, as it has no wireless external communication interfaces. The TARA residual risk value for this product is 116 points, with a weight of 0.05, based on its complexity, innovation level, and connectivity.

B. Discussion

ExCom employs 1,000 developers. For their products and employees, we know that an average developer is capable of fixing the equivalent of 30 points (as introduced in Table II) vulnerabilities per year. The company has the equivalent of 1,000 developer years available. Not more than 5% of development time shall be appointed to fixing vulnerabilities. This results in an available risk pool of 1,500 points:

$$\sum \text{Developers} \cdot \text{Fixing Capability} \cdot \text{Capacity} = 1,000 \cdot 30 \cdot 0.05 = 1,500$$

ExCom has 500 projects each year going into production, equally distributed across its two products. The support period today is the one year warranty period. With these information we come to the inequation as follows:

$$\sum_{i=0}^{250}(29) \cdot 0.02 + \sum_{i=250}^{500}(116) \cdot 0.02 \overset{?}{\leq} \sum 1,000 \cdot 30 \cdot 0.05 \tag{2}$$

$$1,595 \overset{!}{>} 1,500 \tag{3}$$

With Equation (3) showing the Inequality (2) being false, the company in the example is exceeding its available risk pool. A possible solution might be limiting the amount of projects with higher risk, or the expansion of allocated fixing time for the developers, e. g., from $5\,\%$ to $6\,\%$ (Resulting in an available pool of 1,800 points), which would necessitate a price increase of the sold products. If the products need to be supported over a longer period, additional developer capacity is needed and needs to be paid for, e. g., by a maintenance contract with the potential customers. If the introduced new regulations come into effect, the price of the product needs to increase, as the support has to be provided without additional contracts, when dealing with end customers.

This approach might not fit the bill over every period under review, but is supposed to average out over time. When fixing capabilities are vacant, these shall be used to reduce technical debt and refactoring older code bases to reduce the future possibility of found defects or reduce the needed capacity to fix them.

IV. Conclusion

Using the riskpool methodology, we enhanced risk assessment for software. Companies can now transition from abstract numbers to practical tools that guide their risk appetite, including resource management and pre-planning: understand the risk $\rightarrow$ manage the risk. A company is thus able to control the maximum amount of projects that they can handle with their assigned riskpool. For a limited time, additional risks can be taken by stretching the riskpool but this risk can be quantified, not only for single projects but for the sum of all ventures a company has. If the riskpool is depleted, the company can take measures, e. g., by shifting capacities. The weight factor allows to tailor the method to individual products and business cases. Until now, risk estimations were confined to individual projects. With this risk pool approach, companies can gain a comprehensive view of their combined risks across all projects and calculate resilience based on their organization's capacity to address these risks.

REFERENCES

[1] Council of European Union, *Directive 2019/771 of the european parliament and of the council of 20 may 2019 on certain aspects concerning contracts for the sale of goods, amending regulation 2017/2394 and directive 2009/22/ec, and repealing directive 1999/44/ec*, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32019L0771, Accessed: 2024-10-02, 2019.

[2] Deutscher Bundestag, *Gesetz zur umsetzung der richtlinie über bestimmte vertragsrechtliche aspekte der bereitstellung digitaler inhalte und digitaler dienstleistungen [in english: Act implementing the directive on certain aspects of contract law aspects of the supply of digital content and digital services]*, https://www.bmj.de/SharedDocs/Gesetzgebungsverfahren/Dokumente/Bgbl_Digitale_Inhalte.pdf?__blob=publicationFile&v=2, Accessed: 2024-10-02, 2021.

[3] International Organization for Standardization, "ISO/SAE 21434:2021 - Road vehicles – Cybersecurity engineering," Standard, 2021.

[4] International Organization for Standardization, "ISO/IEC 27005:2022 - Information security, cybersecurity and privacy protection – Guidance on managing information security risks," Standard, 2022.

[5] Council of European Union, *Directive 2022/2555 of the european parliament and of the council of 14 december 2022 on measures for a high common level of cybersecurity across the union, amending regulation no 910/2014 and directive 2018/1972, and repealing directive 2016/1148 (nis 2 directive)*, https://eur-lex.europa.eu/eli/dir/2022/2555/oj, Accessed: 2024-10-02, 2022.

[6] International Society of Automation, "ISA/IEC 62443 Industrial communication networks - Network and system security," Standard, 2021.

[7] M. Power, "The risk management of everything," *The Journal of Risk Finance*, 2004.

[8] S. McConnell, *Code complete*. Pearson Education, 2004.

[9] R. N. Charette, "This car runs on code," *IEEE spectrum*, vol. 46, no. 3, p. 3, 2009.

[10] D. McCandless, *Lines of code overview for different software projects, incl. sources*, https://docs.google.com/spreadsheets/d/1s9u0uprmuJvwR2fkRqxJ4W5Wfomimmk9pwGTK4Dn_UI/, Accessed: 2024-10-02.

[11] SecurityScorecard, *Cve details - the ultimate security vulnerability datasource*, https://www.cvedetails.com, Accessed: 2024-10-02.

[12] Synopsis, *Synopsys - black duck open hub*, https://www.openhub.net, Accessed: 2024-10-02.

[13] Common Criteria Working Group, "Common methodology for information technology security evaluation," Technical report, Common Criteria Interpretation Management Board, Tech. Rep., 2017.

[14] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[15] M. Ring, "Systematische security-tests von kraftfahrzeugen [in english: Systematic security testing of motor vehicles]," Ph.D. dissertation, Universität Ulm, 2019.

[16] E. Kovacs, *Cvss scores often misleading for ics vulnerabilities: Experts*, https://www.securityweek.com/cvss-scores-often-misleading-ics-vulnerabilities-experts/, Accessed: 2024-10-02.

[17] E. Kovacs, *Siemens drives rise in ics vulnerabilities discovered in 2022: Report*, https://www.securityweek.com/siemens-drives-rise-in-ics-vulnerabilities-discovered-in-2022-report/, Accessed: 2024-10-02.