Security Debt: Characteristics, Product Life-Cycle Integration and Items

Jabier Martinez, Nuria Quintano, Alejandra Ruiz, Izaskun Santamaria, Iker Martinez de Soria and José Arias
Tecnalia, Basque Research and Technology Alliance (BRTA)
Derio, Spain
{name.surname}@tecnalia.com

***Abstract*—Industries from very diverse domains are realising that security should not be treated in a reactive way (e.g., once the cyberattack has happened). This way, security-related requirements and risks need to be continuously managed, and the need of integrating technical measures should be continuously assessed. In some cases, some decisions led, intentionally or unintentionally, to debt related to security aspects. This security debt is thus incurred when limited approaches or solutions are applied to reach the expected security levels of the system in operation. Identifying and making explicit security debt items is a challenge for companies. In this work, we analyse the literature on security debt to provide initial insights on the topic. Concretely, we discuss its definition, identify its most salient characteristics, present approaches for integrating its management in the product life-cycle, and to present categories and examples of security debt items.**

I. Introduction

Cyber-security is an increasingly important concern given the proliferation of highly and continuously connected networks and systems. From personal devices to cyber-physical systems and critical infrastructures, security measures are required to keep security risks under acceptable thresholds. *Security* is the combination of three criteria: confidentiality, the prevention of the unauthorized disclosure of information; integrity, the prevention of the unauthorized amendment or deletion of information; and availability, the prevention of the unauthorized withholding of information [23]. In addition, security is usually needed to ensure safety (security-informed safety) [24] so in some cases it is also relevant to prevent harms to humans or the environment.

In this work we provide insights on the intersection of *security* aspects with *technical debt*. During the life-cycle of a system (including maintenance) several decisions need to be made by the different disciplines involved in the development. Some of these decisions might incur intentionally or unintentionally in technical debt. "In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible. Technical debt presents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability" [25].

Researchers and practitioners have started to be interested by the term *security debt*, now that technical debt management starts to be more mature as a conceptual framework [20] or even as a new engineering discipline in its own right, at a similar level than risk or assurance management. By analysing the literature on the subject, we aim to respond to the following research questions:

- RQ1: How can we define security debt and which are its main characteristics?
- RQ2: How is security debt management introduced in the product life-cycle?
- RQ3: Which security debt items are described in the current literature?

This paper is structured as follows, Section II presents the methodology for the literature and data analysis on the selected studies. Then Section III presents the results regarding the definition and characteristics (RQ1), Section IV on product life-cycle integration (RQ2), and Section V on the items (RQ3). Finally, Section VI concludes this work.

II. Methodology

Figure 1 illustrates the two phases of the methodology.

Phase 1: Data extraction: We designed a focused query based on searching in the title, abstract, and keywords for security and cybersecurity works related with technical debt. Concretely, the following query was used without any time window in the Scopus digital library[1]:

```
TITLE-ABS-KEY("*security debt"
OR (*security AND "technical debt"))
```

This resulted in 67 entries. The inclusion criteria (inclusive filter) was that the work must be written in English which discarded 4 entries, and conference reviews (summaries of

[1]The first search was conducted in March 2020, and then updated November 2020, with a last check for new entries the 15th December 2020.

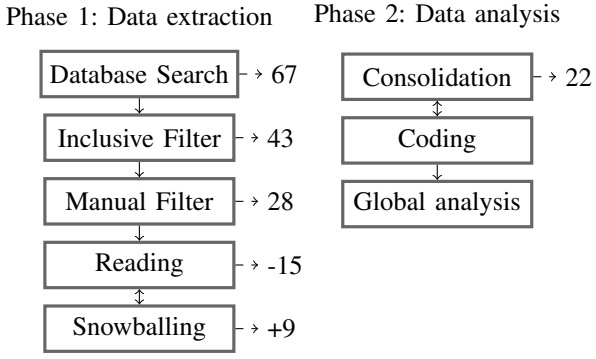

Fig. 1: Methodology including the number of publications which were available in each step.

topics from conferences) were also discarded with 20 entries. The next step was a manual examination by the first author of titles and abstracts (manual filter) which discarded 15 entries as completely unrelated to information systems security (e.g., social security and finance). This way 28 entries were selected for being read and analysed in more detail. This set was used for a snowballing approach [26] where potentially related references found in the papers were recursively added to the literature to analyse. Two authors carried out independent complete reading of each paper (all involved). During the reading of both primary and snowballing entries, 15 instances, that seemed valid in the preliminary manual filter, were confirmed by at least two authors as unrelated to security debt because they were about technical debt but without links to security aspects. We tried to be conservative and we kept publications related to security and technical debt but in those 15 cases no insights were provided. At the end, the total number of relevant publications from which data was extracted were 22. Those originated from the database search after the filters and not excluded after reading were [1]–[17], and the not excluded ones originated from the snowballing were [18]–[22].

Phase 2: Data analysis: The data was analysed to obtain the results. First, a consolidation of the data extracted for each paper was conducted through discussions between the two authors assigned to the paper. This complemented both views and helped to avoid misunderstandings on the content. Once this was finished, workshops among the six authors were conducted to analyse the data and to provide conclusions. The findings were systematically categorized using ids, i.e., the available data was used to perform a coding step (assigning ids). Emergent ids for the three RQs were based on the reading and consolidation steps and then harmonized (e.g., merging similar ids) and refined (e.g., formulated in a more concise way). Finally, a global analysis was performed during the reporting in this paper.

Literature overview: Figure 2 shows the distribution per year of the 22 publications. It suggests an increasing interest on the intersection of security and technical debt. Table I shows the publication fora, i.e., how they are distributed in different journal or venues. Security debt is present in both technical debt and security venues with higher frequency in specialized technical debt venues.

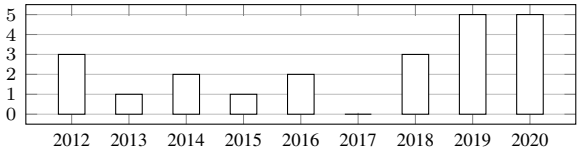

Fig. 2: Distribution of the 22 publications per year

TABLE I: Publication fora

Name	Type	Num
MTD - IEEE Int. Workshop on Managing Technical Debt	Workshop	4
TechDebt - IEEE/ACM Int. Conf. on Technical Debt	Conference	2
JSS, IST, IEEE Software, Journal of Software: Evolution and Process, Enterprise Information Systems	Journal	1 each
ARES, Cyber Sec. and Protection of Digital Services, PROFES, ICSA, ICSE (Doctoral Symposium), SCI, IECON, SEDA, Perf. & Capacity by CMG, EASE, SBSI	Conference	1 each

III. A closer look at Security Debt

We identified certain consensus on the security debt definition even in those that did not explicitly use the term. We present a consolidated definition in Section III-A together with its main characteristics in Section III-B to respond to RQ1.

A. Definition

Security debt refers to the completeness and correctness of the solution that a development team is creating, in terms of the needed security characteristics and requirements for the future operation of the system, in the expected operational context. In other words, security debt is incurred when limited approaches or solutions are applied (intentionally or unintentionally) to reach the needed security levels for the system in operation.

The concept is derived from the technical debt metaphor, however, the interest of the debt is not additional work to maintain the system as in traditional technical debt, but additional security risks that are accumulated throughout the system lifecycle.

The solution being created by the development team can be convenient in the short term to meet deadlines or advance in the project with the current knowledge and skills of the team. This can create a state in which the system might not meet the necessary security characteristics and requirements, compromising the overall quality of the system and the customer expectations on security. Security debt does not only appear during the design activities, it can be incurred since earlier phases such as requirements engineering or risks analyses. The security debt identification and further management allow development teams to know and to record these kinds of situations which appear as a result of business or technical decisions. Also, for business purposes the companies may want to gain visibility and to manage them throughout the whole system lifecycle.

B. Characteristics

We extracted salient characteristics of security debt and its management. We categorized them in Technological, Organizational, and aspects related to its Consequences. Table II summarises the characteristics with a quick reference to the works mentioning or dealing with it.

Technological: It is acknowledged that different artefact types can be the source to incur technical debt [20], and security debt is not an exception [5]–[7], [12], [13]. Different categories can be found in the literature such as requirements [7], code [7], [12], [13], design and architecture [7], [12], [13], configuration [13], environment [12], hardware and physical parts [6], cloud computing infrastructure [13], knowledge distribution [12], documentation [12], [13], and testing [7], [12], [13]. Section V will later present concrete categories and examples of security debt items.

One characteristic that we can highlight from security debt is that it is highly related to security risks. Some authors refer security debt as technical debt containing a security risk [7] or potential security implications [8]. Security engineering techniques (e.g., risk analysis) are used to identify the security debt [6], [8] and security risk in software can be described

TABLE II: Security debt characteristics (RQ1)

Characteristic	References
Technological	
Security debt can be incurred through different artefact types	[5]–[7], [12], [13]
Security debt and security risks are closely tied	[5]–[8]
Security eng. (V&V) helps to identify unintentional security debt	[8]
Continuous security eng. helps to avoid unintentional security debt	[7], [11]
Technical debt can be a source of security debt	[1], [3], [8]
Tradeoffs of security and other quality attributes (e.g., performance) might force to assume security debt	[5], [14]
Organizational	
Organization policies should prioritize security debt	[12], [19]
Security awareness and skills are needed to avoid security debt	[8], [13]
Security debt involves different stakeholders requiring discussions and decision making among them	[5], [14]
Consequences	
Business damage: High interest of the debt	[8], [9], [12], [14], [16], [21]
Interest will be paid mainly when someone exploit the vulnerability	[8], [9], [16], [21]
Paying the principal of the security debt might require to change processes	[16], [19]

in terms of technical debt [8], e.g., including the probability attribute to the security debt item to measure the chances that the security-related defect can be actually exploited [5].

Unintentional security debt should be identifiable thanks to security engineering through validation and verification (V&V) activities [8]. Also, this security engineering should be continuous [7], i.e., not a single-shot effort (e.g., thread modelling only at the beginning of the project) as this may create a quickly obsolete view on the security of the system [11] creating new security issues.

Another characteristic of security debt is that traditional technical debt can be the source of security debt, e.g., suboptimal internal quality in a security critical software component [8]. Under this hypothesis, the correlation between technical debt indicators and software vulnerabilities can identify security debt [1], [3].

The last technological characteristic that we identified is related to tradeoffs among security and other quality attributes that might force the engineers to assume security debt. An example is checking if the optimal system security can be met at the expense of performance [14], or if security and safety decisions are being considered at the same time creating tradeoffs between them [5].

Organizational: It is considered that addressing security debt should be prioritized as the potential damage to the business is high [12], and the usually limited budget should be used for that first [19]. There are also non-technical aspects in technical debt such as personnel capabilities and participation aspects. For security debt, security awareness and skills is relevant to avoid security debt [13]. However, a professional specialized in technical debt management, security engineering and risk management might be hard to find [8].

Security debt might instigate discussions and it might continuously trigger decision making among different stakeholders. A study on open source systems suggests that the case with the highest number of comments when source code wants to be officially integrated is when security-debt-related topics are being discussed [14]. Also, key decision points or check points are defined respectively by NASA and NIST where stakeholders need to take decisions on security and safety, and this was extended with technical debt decisions on these critical considerations [5] making security debt a first-class citizen in the organizational procedures for discussion and decision making.

Consequences: Regarding the consequences of security debt, the first characteristic is that the interest of the debt may get unacceptably high [8] causing important business damages [12], [14], [16], [21]. Security debt is usually measured in terms of loss of business if a vulnerability is successfully exploited rather than, for example, source code maintenance effort in traditional technical debt [9]. Interest will be only paid when someone (intentionally for malicious purposes or unintentionally) exploit the vulnerability [8]. Security debt makes the system more penetrable by malicious actors [21] both in open systems (through externals) and closed systems (through insider threats). In both cases the interest will not be paid if the exposure was not discovered or activated [16]. Paying the principal of the security debt will be usually related to fixing the defect of the vulnerability, however, in other cases, it might imply also to change processes and perform administrative changes [16]. In some cases, changes in processes to perform a proactive security debt management is desired [16], [19].

IV. Security Debt management in the life-cycle

Table III presents the approaches that were identified related to the product life-cycle and the management of security debt.

Across the product life-cycle: Security is a cross-cutting concern affecting all the product life-cycle and the product assets produced in each stage as presented in Section III-B. Thus, all engineering disciplines are subject to generate security debt starting in very early stages such as the concept and requirements phases [10]. The debt management process should then consider all those stages [7] and adjust the techniques aiming to reveal security debt items [8], [13], [14] (e.g., reviews of policies, design, code etc.). Also, ideally security debt cannot be incurred when secure-by-design best practices are followed across the product life-cycle [6]. As this is not yet feasible in several cases, security "touchpoints" during the product life-cycle to avoid security debt are proposed [6].

TABLE III: Security debt in the product life-cycle (RQ2)

Characteristic	References
Across the product life-cycle	
All the stages/disciplines are subject to generate security debt	[6]–[8], [10], [13], [14]
Identification of security debt is a continuous process	[7], [11]
Risk management	
Extending security risk management to manage security debt	[7], [8], [11]
Risk assessment can be used to quantify and prioritize the items	[7]–[9], [12]
Identification techniques	
Security V&V can help to identify security debt	[8]
Assurance can help to identify security debt	[5], [7]
Technical debt indicators can help to identify security debt	[1], [3], [8]
Instances	
SecDevOps	[9]
Cybersecurity framework extended with Security debt	[5]

As mentioned in Section III-B, the identification of security debt should be a continuous process across the product life-cycle [7]. It is not appropriate to think of security as a component that can be added after the product is built, or that it is a quality attribute that can be checked just once. Continuous risk analysis and threats modelling [11] or continuous security testing is thus relevant to identify and monitor security debt.

Risk management: Given the tied relation between security risks and security debt discussed in previous Section III-B, it can be observed how extensions of security risk management to manage security debt were proposed. In [7], they suggest three main processes aligned with ISO/IEC 21827 [27]: security risk identification and assessment, a security engineering process to create risk controls, and a software assurance process. The software development takes as initial input functional, operational, and security requirements based on risks and regulations. In their product life-cycle approach, software development creates debt in all the stages which is taken by a technical debt management process. Apart from this one, a risk-based extension to help in the prioritization of existing debt has been proposed [8] as well as a threat modelling approach extended for security debt analysis [11].

Risk assessment can be used to quantify and prioritize the security debt items [7]. In the previously mentioned prioritization approach [8], quantitative risk assessment measures are suggested. For instance, a combined value of risk probability and impact as proposed in the ISO/IEC 27005 risk management standard [28]. In [9], for security debt prioritization they suggest to take advantage of the Common Weakness Enumeration (CWE) [29] by analysing the technical impacts and the tactics that are used to exploit the weaknesses of the system. In [12], they propose the use of the Common Weakness Scoring System (CWSS) mapping a CWE hierarchy to the Quamoco product quality modelling and assessment approach [30] to help prioritizing items.

Identification techniques: Three product life-cycle processes are identified as the main activities for security debt identification. One is security V&V ranging from reviews and professional observations of the product assets to security testing and simulated attacks [8]. The other one is the assurance process [5], [7] where it must be justified that the implemented solution meets the expected security with appropriate evidences and argumentation (e.g., arguing regulatory compliance through V&V results). An extension of the assurance cases formalism is proposed to explicitly represent the security debt [5]. This way, security debt is integrated at the core of the assurance process which is refined across the product life-cycle. The third one is the technical debt management process. As presented in Section III-B, recent works try to correlate technical debt indicators to security risks [1], [3], [8].

Instances: Finally, we present two concrete instances of how to manage security debt within existing processes. In [9], the authors position their work in the SecDevOps context [31]. DevOps enables to integrate secure development processes into development and deployment processes so security debt can be identified and addressed dynamically. For instance, CWE items detected through static analysis can be mapped to attack tactics [14]. The other instance is cybersecurity frameworks extended with security debt management where each activity in the framework is enhanced with debt considerations [5].

V. Security Debt Items

A security debt item is a concrete work product created during the system evolution where its state makes more difficult the objective to incorporate the needed security characteristics and requirements. Table IV shows the identified item categories and we present concrete examples in the text.

In the product life-cycle, one of the first security debt items that might have cascading effects are deficiencies in the requirements, notably, security requirements might be missing or incomplete [2], [4], [6], [10]. Another item can have as origin requirements which are unrealistic [2] or inadequate [4], e.g., specifying mitigation mechanisms as if they were requirements preventing the engineers of the next product life-cycle stages to take potentially better security-related decisions based on the real requirement [7].

Improper risk management can also be an item [6], for instance uncovered abuse cases (use cases that might create harm to the system) [6], improper penetration testing [6] or improper risk analysis of the physical security [6].

Deficiencies in the design stage might have consequences in security. The architecture should be a means for communication between the requirements and the developers and it should not fail to communicate the security aspects [7]. However, we might have missing security measures in the architecture [6] or improper system analyses creating weaknesses in the architecture [6]. The Common Architectural Weakness Enumeration (CAWE) [32] has been referred in technical debt studies [12] as a relevant catalogue.

Structural deficiencies in the source code have also potential implications on security [7], e.g., potential compromise in code access (private, protected and public access in source code methods) [14], or security flaws in the source code that can be identified with tools [20]. Similarly to architecture deficiencies, the Common Weakness Enumeration (CWE) [29] present an extensive list of violations of good coding practices in the area of security such as SQL injection, cross-site scripting, buffer overflows etc. [21].

Other security debt items are related to compromised lower-level components from suppliers/third-parties [7], e.g.,

TABLE IV: Security debt items (RQ3)

Items	References
Deficiencies in security requirements	[2], [4], [6], [7], [10]
Deficiencies in security abuse cases identified during risk identification	[6]
Defective architecture and its implications on security	[6], [7], [12]
Structural deficiencies in source code and its implications on security	[7], [14], [20], [21]
Compromised lower-level components from suppliers/third-parties	[7]
Defects in data integrity (database) and its implications on security	[17]
Defects in physical parts (e.g., CPS) and its implications on security	[6]
Deficiencies in security validation and verification activities	[6]
Limitations in the conformity to standards, guidelines etc.	[5]

dependency debt with security risks that can be exploited through supply chain attacks. How the database is implemented can also create security debt items such as defects in data integrity [17]. Defects in physical parts (e.g., of a Cyber Physical System or IoT device) can also present security vulnerabilities [6] (e.g., glitching electrical attacks). Incorrect access control of an intruder or unauthorised personnel through the physical parts of the system [6] is also a debt item where hardware plays an important role.

Finally, we have items related to deficiencies in security V&V activities such as missing code reviews [6], or limitations in the conformity to standards and guidelines, e.g., in the health domain a specific device communication standard and a safety standard are mentioned regarding this aspect [5].

VI. Discussion and conclusions

We have shown that security debt is a trend with several ongoing research and new approaches. It deals with a real-life problem posing fundamental challenges to industry and with potential direct consequences for the final users and society.

We provided an initial overview of the security debt topic through a review of the literature focused on this specialized intersection between technical debt and security management. We discussed its definition and more importantly, we summarized its main characteristics, how it was introduced in the product life-cycle, and which types of security debt items have been reported. Direct references to each of these individual aspects were provided. We consider that those insights and clarifications are needed for the community for an aligned vision in next research works of this incipient field.

As identified gaps in the workshops among the authors of this paper, we can mention the lack of 1) detailed industrial experience reports and datasets, 2) tool-supported approaches for early product life-cycle stages, and 3) analyses of security-by-design on security debt. As further work, an approach to quantitatively manage security debt starting at the requirements engineering and risk management disciplines will be designed based on the aspects identified in this work. This will include extensions of risk management and assurance tools.

Acknowledgment: Thanks to the TRUSTIND project (Creating Trust in the Industrial Digital Transformation), an Elkartek project funded by the Basque Government.

References (Data analysis entries)

[1] R. Halepmollası, "A composed technical debt identification methodology to predict software vulnerabilities," in *ICSE-Companion*, 2020.

[2] M. Zarour, M. Alenezi, and K. Alsarayrah, "Software security specifications and design: How software engineers and practitioners are mixing things up," in *EASE*, 2020.

[3] M. Siavvas, D. Tsoukalas, M. Jankovic, D. Kehagias, and D. Tzovaras, "Technical debt as an indicator of software security risk: a machine learning approach for software development enterprises," *Enterprise Information Systems*, 2020.

[4] W. Behutiye, P. Karhapää, L. López, X. Burgués, S. Martínez-Fernández, A. M. Vollmer, P. Rodríguez, X. Franch, and M. Oivo, "Management of quality requirements in agile and rapid software development: A systematic mapping study," *IST*, vol. 123, p. 106225, Jul. 2020.

[5] X. Larrucea, I. Santamaria, and B. Fernandez-Gauna, "Managing security debt across PLC phases in a VSE context," *Journal of Software: Evolution and Process*, vol. 32, no. 3, 2020.

[6] B. Brenner, E. Weippl, and A. Ekelhart, "Security related technical debt in the cyber-physical production systems engineering process," in *IECON. Conf. of the IEEE Industrial Electronics Society*, vol. 1, 2019.

[7] K. Rindell, K. Bernsmed, and M. G. Jaatun, "Managing security in software," in *ARES*, 2019.

[8] K. Rindell and J. Holvitie, "Security risk assessment and management as technical debt," in *2019 Int. Conf. on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE, Jun. 2019.

[9] C. Izurieta and M. Prouty, "Leveraging SecDevOps to tackle the technical debt associated with cybersecurity attack tactics," in *Int. Conf. on Technical Debt (TechDebt)*. IEEE, May 2019.

[10] G. Robiolo, E. Scott, S. Matalonga, and M. Felderer, "Technical debt and waste in non-functional requirements documentation: An exploratory study," in *Product-Focused Software Process Improvement*, 2019.

[11] L. Sion, D. V. Landuyt, K. Yskout, and W. Joosen, "SPARTA: Security & privacy architecture through risk-driven threat assessment," in *Int. Conf. on Software Architecture Companion (ICSA-C)*, Apr. 2018.

[12] C. Izurieta, D. Rice, K. Kimball, and T. Valentien, "A position study to investigate technical debt associated with security weaknesses," in *Int. Conf. on Technical Debt - TechDebt*, 2018.

[13] M. M. Kumar and A. N. Nandakumar, "Exploring multilateral cloud computing security architectural design debt in terms of technical debt," in *Smart Computing and Informatics*, 2018.

[14] M. C. Silva, M. T. Valente, and R. Terra, "Does technical debt lead to the rejection of pull requests?" in *SBSI: Volume 1*, 2016, p. 248–254.

[15] D. Russo, "Benefits of open source software in defense environments," in *Int. Conf. in Softw. Eng. for Defence Applications*, 2016, pp. 123–131.

[16] D. P. Kalm and J. Rhodes, "Technical debt - the cost in performance and security," Int. Conf. on Performance and Capacity 2014 by CMG URL: https://share.confex.com/share/123/webprogram/Session15955.html, 2 2014.

[17] J. H. Weber, A. Cleve, L. Meurice, and F. J. B. Ruiz, "Managing technical debt in database schemas of critical software," in *Int. Workshop on Managing Technical Debt*, Sep. 2014.

[18] K. Power, "Understanding the impact of technical debt on the capacity and velocity of teams and organizations: viewing team and organization capacity as a portfolio of real options," in *MTD workshop*, 2013.

[19] J. Letouzey and M. Ilkiewicz, "Managing technical debt with the SQALE method," *IEEE Software*, vol. 29, no. 6, pp. 44–51, 2012.

[20] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *J. Syst. Softw.*, vol. 101, 2015.

[21] B. Curtis, J. Sappidi, and A. Szynkarski, "Estimating the size, cost, and types of technical debt," in *MTD workshop*, 2012, pp. 49–53.

[22] J.-L. Letouzey, "The sqale method for evaluating technical debt," in *Int. Workshop on Managing Technical Debt*, ser. MTD '12, 2012, p. 31–36.

References

[23] A. Abdulkhaleq, S. Wagner, D. Lammering, H. Boehmert, and P. Blueher, "Using STPA in compliance with ISO 26262 for developing a safe architecture for fully automated vehicles," in *Automotive - Safety & Security*, 2017.

[24] K. Netkachova and R. E. Bloomfield, "Security-informed safety," *IEEE Computer*, vol. 49, no. 6, pp. 98–102, 2016.

[25] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. B. Seaman, "Managing technical debt in software engineering (dagstuhl seminar 16162)," *Dagstuhl Reports*, vol. 6, no. 4, pp. 110–138, 2016.

[26] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *EASE '14*, 2014.

[27] ISO/IEC, "Standard 21827:2008 Information Technology – Security Techniques – Systems Security Engineering – Capability Maturity Model (SSE-CMM)," 2008.

[28] ——, "Standard 27005:2018, Information technology — Security techniques — Information security risk management," 2018.

[29] MITRE, "The Common Weakness Enumeration (CWE)," 2006. [Online]. Available: http://cwe.mitre.org

[30] S. Wagner, K. Lochmann, L. Heinemann, M. Kläs, A. Trendowicz, R. Plösch, A. Seidi, A. Goeb, and J. Streit, "The quamoco product quality modelling and assessment approach," in *ICSE*, 2012.

[31] A. A. U. Rahman and L. Williams, "Software security in devops: Synthesizing practitioners' perceptions and practices," in *Int. Workshop on Continuous Software Evolution and Delivery (CSED)*, 2016.

[32] J. C. S. Santos, K. Tarrit, and M. Mirakhorli, "A catalog of security architecture weaknesses," in *ICSA Workshops*, 2017.