Security Risks in the Software Development Lifecycle: A Review

Article · March 2023

DOI: 10.30574/wjaets.2023.8.2.0101

3 authors:

David Odera
Tom Mboya University
7 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Martin Otieno
JOOUST
5 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Jairus Ounza
Kabarak University
4 PUBLICATIONS **0** CITATIONS
SEE PROFILE

World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(REVIEW ARTICLE)

Check for updates

Security risks in the software development lifecycle: A review

David Odera [1,*], Martin Otieno [2] and Jairus Ekume Ounza [3]

[1] *Tom Mboya University, Homa-Bay, Kenya.*
[2] *Jaramogi Oginga Odinga University of Science and Technology, Bondo, Kenya.*
[3] *Kabarak University, Nakuru, Kenya.*

World Journal of Advanced Engineering Technology and Sciences, 2023, 08(02), 230–253

Publication history: Received on 23 February 2023; revised on 03 April 2023; accepted on 06 April 2023

Article DOI: https://doi.org/10.30574/wjaets.2023.8.2.0101

Abstract

Software security is one of the most critical concerns in modern software development, especially in safety-critical systems whose failure can lead to environmental damage, substantial property, or loss of human lives. In addition, flawed applications have been shown to exhibit unpredictable behavior while software products with numerous vulnerabilities present attack vectors that can be exploited by attackers. To address some of these problems, vulnerability prediction has been deployed for early detection of security risks in the software development lifecycle (SDLC). This can potentially facilitate decision making during the SDLC, resulting in the production of more secure software. Prioritizing security during SDLC permits developers and stakeholders to identify and resolve possible security concerns early on in the process. The aim of this paper is therefore to offer some in-depth review of software systems security issues. In addition, the various measures that have been put in place to mitigate security issues during SDLC are discussed.

Keywords: Software; Security; SDLC; Vulnerabilities; Attacks

1. Introduction

The concept of being free from harm or threat constitutes security [1]. On the other hand, software security deals with the utilization of various techniques and methods to analyze defend and mitigate effects of vulnerabilities that may be inherent in software systems. Therefore, security is an essential component of the software application [2]. This is more pronounced in safety-critical systems, which are systems whose failure may result in environmental damage, substantial property, or endangerment or loss of human lives [3]. Therefore, it is critical that reliability confidentiality, and integrity of software used in these systems be secured. However, the authors in [4] explain that software security and durability (SSD) in the development cycle of software have continued to present new challenges for developers. In this perspective, various factors characterize security [5] and durability include usability, dependability, trustworthiness and human trust. As explained in [6], security and durability collectively depict convenience and there is always a trade-off between these two concepts. The evaluation of SSD is a dynamic process and as such, multi-criteria decision making (MCDM) strategy is always utilized. The authors in [7] explain that MCDM techniques can be deployed in several contexts, such as software and frameworks. Here, MCDM procedures allow the software developers to choose options among various conflicting choices.

Software security is one of the most critical concerns in modern software development [8]. This is because flawed applications have been shown to exhibit unpredictable behavior. In addition, software with many vulnerabilities presents attack vectors that can be exploited by malicious entities. As pointed out in [9], the efficient utilization of software in quantum computing calls for security enhancements so that new security threats can be neutralized. The authors in [10] point out that software security and its durability in quantum computer existence will improve human

*Corresponding author: David Odera

trust and buyer's dependability. As explained in [11], the advent of quantum computing together with improvements in programming have generated the necessity of building effective security mechanisms [12] in the initial stages of software development itself. However, there is always trade-offs between ease-of-use and security. Another significant aspect of software systems is trust, which deals with the normal operation of the software product while preserving information security and optimal usability [13].

As explained in [14], software developers encounter several challenges that impede their operations. Such issues include limitations in improvements due to high costs, time-to-market necessities, profitability sway and consumer loyalty concerns. All these challenges lead to compromise in secure programming. Basically, market demands call for frequent software releases. However, traditional software development methods tend to be sequential and rigid, and hence are not well-suited for the demands of the market [15], [16]. In addition, modifications to software due to frequent changes in software requirements are time consuming and costly, hence cannot be effectively [17] handled by the traditional methods [18]. Most of the small and medium-sized Enterprises (SMEs) focus on providing specific solutions for particular needs of their customers. As such, they normally base their software development methods on agile principles [19]. Due to highly competitive pressure faced by such enterprises, their solutions must rely on highly adaptable and cost-efficient frameworks which offer solutions that meet their needs. As discussed in [20] and [21], agile software development methods such as Scrum and eXtreme Programming have been developed to overcome the shortcomings of traditional methods. Most of the agile methods lean on software development iterations and increments [22], self-organizing teams, face-to-face daily communications among team members, and short feedback loops with customers [23], [24]. These features are among the key reason why agile methods offer a highly adaptable, efficient [25], [26], [27], and fast software development process. They have therefore become the most popular paradigms for software development today [28].

Despite the fact that the developers invest many resources in resolving security concerns throughout the initial phase of software development, no consideration is given to the life span of the software [29]. Although there is a growing demand for secure software, developers are encountering new challenges in meeting users' demands while developing the product. To this end, this paper makes the following contributions:

- Various sources of software quality and security issues are described in detail
- Rationale for the increasing need for software security is discussed;
- Numerous techniques and approaches to software security enhancement are highlighted.

The rest of this article is structured as follows: Part 2 discusses the Sources of software quality and security issues while Part 3 describes the need for software security. On the other hand, Part 4 presents the techniques and approaches to software security while Part 5 highlights some of the research gaps. At the end, Part 6 concludes this article and gives some future research scope.

2. Sources of software quality and security issues

There are numerous sources of software vulnerabilities, compromises, threats and attacks. the authors in [30] explain that the primary reason why software development teams do not implement security is due to a lack of knowledge and experience in different types of vulnerabilities. This is supported by the authors in [31], who point out that datamany competent software development teams still do not implement secure, privacy-preserving software, even though techniques to do so are now well-known. The major cause of this is lack of priority and resources for security. In addition, security and performance (SAP) verification operations are often neglected when the software development project's timeframe or budget is shortened. This leads to reduction in the system's quality since the product owner may terminate the verification process before all activities are completed, reducing the verification coverage [32], [33]. On the other hand, a strong correlation has been found in [34] between the financial records of the software development enterprises (such as sales and financial performance) and the number of vulnerabilities that their products may contain. As discussed in [35], organizations' security efforts are less effective when developers perceive a disinterest in adopting software security practices. This usually occurs when there are no perceived negative consequences to the customers or the business from the lack of security in the SDLC.

Software performance issues are often introduced by engineers who are unaware of their existence. In some cases, problems are not obvious because they are not caused by the code itself but by how the code responds to something else [36]. No matter how enthusiastic a software development team may be about security, if they do not have appropriate knowledge, time and resources (both financial and otherwise) to make their software secure, they are unlikely to be effective at achieving it [37]. On their part, authors in [38] explain that organizations usually employ traditional training resources and methods that developers do not feel are practical and actionable. For instance, most

of these learning resources focus on policies and protocols [39], reading, watching videos, or office conversation by either internal teams or external parties. In some cases, secure software development can be inspired by other security-centric development contexts [40]. Some literature has pointed out that software developers are responsible for many of the software vulnerabilities which occur when developers face pressure to meet customer requirements and deliver features quickly. In most cases, developers treat security as a non-functional requirement that is less critical than delivering features, unless employers or application users impose security compliance [41]. Unfortunately, the delayed consideration of security issues renders it more challenging and expensive to address them in later stages [42].

Technical Debt (TD) is one interesting software related factor that may indicate software security risks. TD is used to quantify long-term software quality problems which are caused by quality compromises that provide short-term benefits. Basically, it is used to quantify the effort that is required for fixing design and code quality issues (such as code smells and violations of coding rules and best practices), which are introduced by the developers due to sacrifices they make to the quality of the code they produce, normally in an attempt to meet strict production deadlines. Therefore, a large TD value indicates that the corresponding software product contains an increased number of quality issues, which in turn indicates poor overall quality. As such, several researchers have started theoretically examining the feasibility of using TD as an indicator of security risk [43], [44]; [45] [46], [47],[[48]. On the other hand, lack of security culture in teams and organizations has been identified in [49] and [50] as a significant deterrent to the adoption of security. In addition, organizations that do not provide the necessary resources have been noted to prevent developers from implementing security [42]. For instance, when managers see security as a resource conflict with feature development, developers also perceive implementing security as not worth the time and energy.

Based on some established standards security can be one of the key features of any software product [51]. Here, software is considered secure when it is adequately resistant to the alterations during intentional or unintentional attacks [52], [53]. Although heavily utilized, agile software development method is poorly suited for secure software development [54]. This is because they predominantly focus on functional requirements of the developed software while security is often neglected [55], [56]. On the other hand, the specificities of SMEs are pronounced in enterprises practicing the DevOps concept, since the DevOps teams are customarily heavily loaded. Such load usually results in software deliverables' degradation, quality decrease, and minimization of the efficiency of the DevOps teams [57]. Several studies have pointed out the need to investigate the behavioral aspects of security adoption. In particular, developers' motivations and attitudes towards security [49], [50], [58], [35] is a hot topic. However, despite the availability of these many resources, developers continue to introduce security vulnerabilities in source code, and organizations lack proper guidelines for designing strategies to mitigate poor security. For instance, despite receiving a tailored security checklist as a reminder during code reviews, developers are unable to find more vulnerabilities than when are just instructed to focus on security issues [59]. Table 1 presents a summary of these sources of software quality and security issues.

Table 1 Sources of software quality and security issues

Author (s)	Software quality and security sources
[30]	Explain that the primary reason why software development teams do not implement security is due to a lack of knowledge and experience in different types of vulnerabilities.
[31]	Point out that datamany competent software development teams still do not implement secure, privacy-preserving software, even though techniques to do so are now well-known.
[32], [33]	The authors observed that, apart from lack of priority and resources for security, security and performance (SAP) verification operations are often neglected when the software development project's timeframe or budget is shortened, leading to reduction in the system's quality since the product owner may terminate the verification process earlier, reducing the verification coverage.
[34]	Noted a strong correlation between the financial records of the software development enterprises (such as sales and financial performance) and the number of vulnerabilities that their products may contain.
[35]	Found out that organizations' security efforts are less effective when developers perceive a disinterest in adopting software security practices whenever they perceive no negative consequences to the customers or the business from the lack of security in the SDLC.
[36]	Observed that software performance issues are often introduced by engineers who are unaware of their existence.

[37]	Found out that a software development team, without appropriate knowledge, time and resources (both financial and otherwise) to make their software secure nay not achieve its security.
[38]	Explain that organizations usually employ traditional training resources and methods that developers do not feel are practical and actionable.
[39]	The author states thatmost of the learning resources focus on policies and protocols, reading, watching videos, or office conversation by either internal teams or external parties
[40]	State that secure software development can be inspired by other security-centric development contexts.
[41]	Found out that, in most cases, developers treat security as a non-functional requirement that is less critical than delivering features, unless employers or application users impose security compliance
[42]	Pointed out that the delayed consideration of software security issues renders it more challenging and expensive when addressed in later stages.
[43], [44]; [45] [46], [47], [48].	Examined the feasibility of using TD as an indicator of security risk.
[49] and [50]	Identified lack of security culture in teams and organizations as a significant deterrent to the adoption of security.
[51].	Stated that, based on some established standards security is a key features of any software product.
[52], [53]	Conclude that software is considered secure when it is adequately resistant to the alterations during intentional or unintentional attacks.
[54].	Noted that, although heavily utilized, agile software development method is poorly suited for secure software development
[55], [56]	Observed that focus on functional requirements of the developed software is predominant while security is often neglected.
[57]	Found out that the specificities of SMEs are pronounced in enterprises practicing the DevOps concept, since the DevOps teams are customarily heavily loaded, resulting in software deliverables' degradation, quality decrease, and minimization of the efficiency of the DevOps teams.
[49], [50], [58], [35]	Pointed out the need to investigate the behavioral aspects of security adoption, in particular, developers' motivations and attitudes towards security.
[59]	Found that despite receiving a tailored security checklist as a reminder during code reviews, developers are unable to find more vulnerabilities than when are just instructed to focus on security issues.

3. The need for software security

Software security entails the usage of methods and techniques to analyze, mitigate, and defend software systems against vulnerabilities. These methods guarantee that software continues to operate and is protected against threats [60]. Security considerations must be made at every level of the software development process, where the primary objective is to find faults and problems as early as feasible. Security and performance (SAP) are two critical Non-functional requirements (NFRs) that affect the successful completion of software projects [61]. Therefore, organizations need to follow the practices that are vital to SAP verification. These practices must be incorporated into the software development process to identify SAP-related defects and avoid failures after deployment. Although security and performance are two different NFRs, they are in some way related to each other. Essentially, security is a performance indicator that is affected by threats which impact the performance of individual components of software during service rendering [62], [63]. In a nutshell, both security and performance are indicative of the software's efficiency, implying that performance and security are indicators of the software's development level.

As discussed in [64], it is critical to use secure channels to share valuable information since it is not always possible to ensure the trustworthiness of individuals or groups. This is crucial when working with unfamiliar people or entities. The exploitation of a single vulnerability may lead to far-reaching consequences, including financial losses and

reputation damages. In this scenario, blockchain systems comprising of data protection solutions can be used to safeguard data against attacks. For instance, a blockchain-based data provenance framework for the cloud has been proposed in [65]. This framework is based on the identified challenges concerning security and performance in adopting the proof-of-work (PoW) consensus protocols for data attribution on a cloud-based platform. The relevance of software systems to modern civilization generates exceptional concerns about several essential software quality attributes, which are described as Non-functional requirements (NFRs). Here, security, performance, reliability, maintainability, usability, and scalability are all defined by NFRs [66], [67]. All these NFRs constrain the system's design across backlogs and are vital as functional requirements. This is because they ensure that the system is useful and effective [68]. Systems that fail to fulfill any of these requirements may fail to meet internal business, user, or market expectations [69]. In addition, these NFRs may result in substantial legal penalties if they are not adhered to.

TD has been established to be closely related to the maintainability quality attribute in [70], [71], [72], [73], especially in distributed software applications. These applications are one of the most important applications currently deployed. The increasing demand has led to a rapid increase in the number and complexity of distributed software applications. However, such applications are more vulnerable to different types of attacks due to their distributed nature [74], [75]. Code smells are code or design patterns that often violate one, or more than one programming principle [76]. Therefore, they potentially cause deeper problems in further development and maintenance of the software. As discussed in [77], various studies have explored the involvement of code smells in the presence of TD. The code smell problems can impede the software maintenance process and impose the need for code refactoring. In addition, the various static code analyzers used to identify TD through source code analysis aim to locate bugs or violations that can cause quality decay [78], [79], [80]. These code-level issues need to be eliminated typically through the application of code refactoring. The authors in [81] and [82] point out that distributed systems have superior reliability, availability and incremental scaling potential compared to centralized systems. Essentially, component based distributed systems deploy constituent components across multiple hosts and regions to deliver services over the internet. Unfortunately, validating the security of such applications is challenging because they are exposed to different kinds of attacks [83], [84]. Table 2 gives a summary of the various needs for software security.

Table 2 Need for software security

Author (s)	Need for software security
[60]	Guarantee that software continues to operate and is protected against threats
	Find faults and problems as early as feasible
	Identify Security and performance related defects and avoid failures after deployment
	Ensure security; performance indicators of individual components of software is not affected by threats during service rendering
[64]	Ensure safe channels to share valuable information
	To prevent exploitation of vulnerabilities that may lead to far reaching consequences that may include financial losses and reputation damages
[65]	Address identified challenges concerning security and performance in adopting the proof-of-work (PoW) consensus protocol for data attribution on cloud-based platform
[68]	As a non-functional requirement security ensures that the system is useful and effective
[69]	Fulfilment of internal business, user or market expectations
[74] [75]	To address vulnerabilities and attacks due to complexity of distributed applications
[78] [79] [80]	To address issues related to bugs and violations that can cause software decay
[83] [84]	To enable validation of component based distributed systems distributed across multiple hosts and regions to deliver services over the internet

4. Techniques and approaches to software security

The researchers in industry and academia have developed numerous techniques to enhance software security. For instance, adherence to some software development method, such as using the right practices, tools, and techniques can makes the process of developing software more manageable and efficient, resulting in greater competitiveness and success [85]. However, some researchers have recognized the importance of awareness of the social perception of security adoption. For instance, when the whole team is responsible for security, the motivation for adopting and implementing security could have a snowball effect and lead to motivating more team members to acknowledge the value of adopting security [41], [35]. There is also need for software systems to adhere to SAP [86]. This can be achieved by having the software development companies incorporating quality assurance activities throughout the product life

cycle to analyze these qualities, hence avoiding SAP difficulties after software release. Based on a task-based study, the authors in [87] established that developers tend to see only the activity of writing code to be security-relevant. Therefore, the authors suggested a need for a stronger focus on the tasks and activities surrounding coding. On their part, the authors in [75] have developed a new mechanism that uses blockchain technology [88] to devise a security testing mechanism to detect attacks on distributed software applications. This technique can detect several categories of attacks, such as denial-of-service attacks, malware and others.

As explained in [89], vulnerability prediction can enable early detection of security risks in the software development lifecycle (SDLC). The focus here is on analyzing the ability of particular software related factors such as software metrics to detect vulnerabilities in software. It also involves the development of vulnerability prediction models based on these factors [90], [91]. This process facilitates decision making during the SDLC, leading to the production of more secure software. On the other hand, the authors in [92] have developed a comprehensive empirical evaluation of automated security analysis tools for detecting vulnerabilities. Other studies have pointed out the need for the business alignment of software security. For instance, the authors in [93] have used case studies to explain the need for the alignment of security goals [94] with business goals. Similarly, the authors in [95] have identified a frequently used approach for developer teams of 'product negotiation', which advocates for the involvement of product managers and other stakeholders in security discussions. On the other hand, the studies in [96], [97], [98], [99], [100], [101] have proposed using blockchain technology to build a security testing mechanism [102] to detect attacks, more so those targeting distributed software applications.

The authors in [103] and [104] have explained that security should be a priority throughout the SDLC. This allows developers and stakeholders to identify and resolve possible security concerns early on in the process.As such, it is necessary to adopt the concept of secure software development lifecycle (SSDLC), as shown in Figure 1. In this regard, SSDLC involves the incorporation of security best practices into an existing SDLC for achieving secure software. This demands concerted effort at each level of the SDLC, from requirement collection through deployment and maintenance [105]. In this way, security vulnerabilities are addressed in the SDLC pipeline long before deployment to production with a dedicated effort. This serves to minimize the possibility of detecting security vulnerabilities in the deployed software and seeks to lessen the consequences if detected [106].Basically, the goal of SSDLC is not to replace conventional security checks but rather to integrate security into software development duties and empower them to build safe applications from the start. As shown in Figure 1, organizations need to adopt SSDLC practices during development. This includes security requirement analysis, security design, security implementation, security testing, as well as operation and maintenance. Once software development is complete, security verification must be carried out before release. This may involve the application of essential security verification techniques.

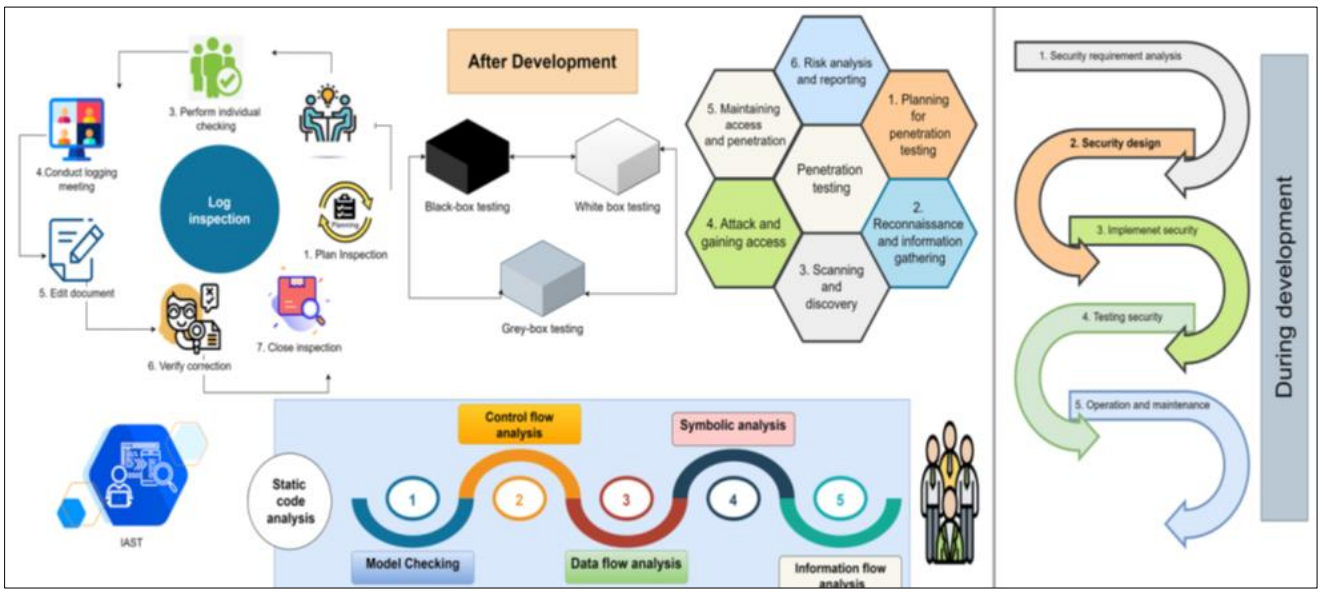

Figure 1 Security verification [61]

In a nutshell, security verification deals with the technical validation of the software application before its deployment.Its goal is to identify security breaches and to ensure that the developed application meets security requirements [107], [108], [109]. In order to establish security requirements, there is need for organizations to first

understand the threat and risk modeling techniques and process. Figure 2 presents the threat modeling process, which is geared towards better identification of risks and their corresponding severity.

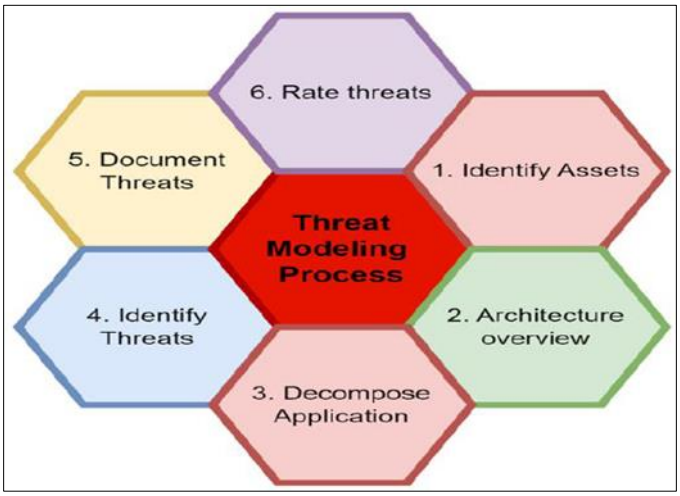

Figure 2 Threat modeling process [61]

The commonly used security verification techniques include log inspection, penetration testing, static code analysis and interactive application security testing (IAST). Here, log inspection encompasses the systematic process that consists of seven activities, as shown in Figure 1. These activities include planning the inspection, holding kick-off meetings, performing individual checking, conducting log meetings, editing the document, verifying corrections, and inspection closure [110]. On the other hand, penetration test is normally performed using a failure-based approach, with test cases designed to investigate known flaws found in popular security vulnerability repositories. Another penetration test approach is experience based, in which a security expert plays the role of a malicious user attempting to access the system [111], [112]. As evidenced in Figure 1, this is a systematic process consisting of six steps. This penetration testing is executed either manually or with the help of automated tools such as Arachni, SQL Injection ME, OWASP ZAP, XSS ME, Meta Exploit, NMAP, Burp Suite, and Whatweb.

Authors in [113] explain that static code analysis (SCA) before executing a program is a debugging method in which one must compare a code set to one or more sets of coding rules. It essentially resolves source code flaws that might lead to vulnerabilities. This analysis can be performed manually through the examination of each line of code or automatically using automated techniques. In most cases, SCA is performed in an automated using tools such as HP Fortify SCA and Brakeman. On the other hand, IAST solutions help organizations to uncover and manage security risks associated with runtime application vulnerabilities. IAST uses software instrumentation to monitor an application's behavior and performance. These solutions put agents and sensors in operating applications and analyze all application interactions to find vulnerabilities in real-time. In this environment, vulnerability prediction facilitates the development of secure software through the identification and mitigation of security risks early enough in the software development lifecycle [114]. On the other hand, other researchers have suggested the application of blockchain technology and specific criteria to address data-related issues, such as dependability, security and trust [115], [116].

Some researchers have identifies four essential drivers that influence developers in the adoption of security practices. These drivers include provision of security education; having non-technical skills;creation and participation in communities of practice; and fostering hands-on learning, self-teaching, and osmosis. For example, the authors in [117] have pointed out that security-related workshops facilitated by managers are more effective compared with those facilitated by developers or security specialists. On the other hand, the authors in [118] have evaluated whether security patterns might be an effective intervention to improve secure development in teams of student software developers. Although the results suggested a benefit, they were statistically inconclusive. On their part, the authors in [119] have pointed out the importance of blockchain for IoT security, privacy and management. An architectural framework for analyzing the performance deterioration of software caused by security measures has been developed in [120]. Basically, this framework introduces a collection of UML models that describe security mechanisms which may be used with performance-annotated UML application models to build SAP-critical systems.

According to [121], traditional approaches to security prioritization include business impact analysis, continuous aligning of business and security goals, security goals' identification [122], [123] on predominantly strategic levels,

security triage [124], introduction of more security controls [125], security intention recap meetings [126], and the identification of architectural security requirements on the operational level. Using blockchain technology and smart contracts, authors in [92] have performed an empirical evaluation of automated security testing tools to detect security vulnerabilities. To encourage developer security, there is a need to raise developers' security awareness [127]. This was achieved using playful workshops [128]. However, awareness is only the first step [129], as individuals need to be supported through training to have the ability to perform the expected behavior. As pointed out in [41], developers feel motivated to adopt security practices when they are aware of similar software (to the one they work on) suffering a security breach [130]. This is because this situation becomes an eye-opener for them. To improve the security of the network, authors in [131] and [132] have deployed the quantum key movement approach. On the other hand, a DistBlockNet distributed IoT network architecture using blockchain has been developed in [133] to offer scalability, flexibility and communication security between different smart device categories.

A taxonomy of twenty assurance techniques has been developed in [134] based on a survey of security specialists, while the relevance of having interpersonal skills such as communication skills for dealing with all stakeholders involved in a security issue has been emphasized in [135]. According to [136], the key to enabling good security behavior [137] is good motivators. This may include feedback, situations or rewards that encourage the behavior. On their part, the authors in [138] have investigated whether open-source software products' popularity can be used as an indicator of their security level. The results indicate that popularity may not constitute a reliable security indicator. A study in [139] offers support for developers and tool recommendations, containing much valuable practitioner experience, but little objective assessment of the advice provided. On the other hand, seven considerations(categories) that organizations and stakeholdersneed to pay attention to in order to foster the adoption of software security practices by developers have been highlighted in [8]. Figure 3 shows this list of these seven categories.

However, at time individuals may feel that they are not well equipped for security or, disillusioned to the benefit of promoting security. When this happens, motivators will be perceived as a nuisance and may reinforce archetypal behaviors [140]. On the other hand, traditional activities associated with risk management such as risk identification, risk analysis, risk assessment, and their variations have been proposed in [141], [142], [143]. Knowledge sharing has been noted to be crucial for learning security practices. For example, developers can learn the best from talking to other people in their teams as they can learn more technical skills while applying existing knowledge [38]. Similarly, peer-based learning has been identified as an effective method to learn security practices [144]. According to [117], developers usually perceive mentoring as an effective way to understand the rationale behind threats and techniques to mitigate them. However, no previous studies have pointed out the essential role of using security practices as learning tools. The authors in [145] have explained that fuzzy analytic hierarchy process (AHP) procedures offer assessment and weighting of the security factors in the selected software security.

Figure 3 Consideration to foster the adoption of software security practices [8]

In order to better understand SSDLC, there is need to analyze the existing SSDLC models. These models include System security engineering capability maturity model (SSE-CMM), Microsoft security development lifecycle (MS- SDL) and Software assurance maturity model (SAMM). Here, SSE-CMM denotes a process-oriented paradigm used to create safe systems and is divided into processes and stages of maturity. Generally, the procedures specify what the security

engineering process must do, and the maturity levels classify how well the process achieves its objectives. On its part, Microsoft's SDL incorporates stringent security standards, industry-specific tools, and required procedures into the creation and maintenance of all software products. However, SAMM is an open framework designed to assist businesses in developing and implementing a software security strategy appropriate to the unique threats they face. Table 3 provides a comparison of these models.

Table 3 Comparison of existing SSDLC

Security Practices	**SSE-CMM**	**MS-SDL**	**SAMM**
Threat modeling	√	√	√
Security architecture	√	√	√
Physical security	√	×	×
Following all applicable laws, policies,and procedures	√	×	√
Logical security	√	×	×
Risk analysis	√	√	√
Definition of security requirements	√	√	√
Secure configuration management	√	×	×
Source code analysis	×	√	√
Security verification	√	√	√
Security training and awareness	√	√	√
Vulnerability analysis	√	√	√
Secure design	√	√	√
Secure development techniques and applications	√	√	√
Secure delivery	√	×	√
Vulnerability management	√	√	√
Security in an active operating environment	√	√	√
Secure integration with peripheral	√	√	√

As explained in [146], regression models such as logistic regression have demonstrated sufficient performance in vulnerability prediction. On the other hand, some attempts have been made in [90] to built text mining-based vulnerability predictors with sufficient predictive performance. According to [147], there is a shift towards adopting deep learning to improve the predictive performance of text mining-based vulnerability predictors [147]. The ability of software metrics to be utilized as vulnerability indicators (predictors) has been supported by numerous studies [148], [149], [150], [151]. Along with common complexity [152], coupling, and cohesion (CCC) metrics, these studies investigated additional software metrics. As explained in [153], another viable solution is the altering of the product backlog through the introduction of security-related user stories, such as abuser stories and generic security user stories. These are basically security-related user stories, which is a set of possible scenarios and threats to the end product.

Recently, they have been attempts in examining the feasibility of quantifying software security indirectly through the notion of TD. The authors in [44], [46] have offered guidelines on how the concept of TD can be extended to support software security. On the other hand, the authors in [45] and [48] have presented ways for prioritizing security bugs as TD items (quality issues). However, these studies only offer theoretical evaluation of the feasibility of TD [154]-[158] to be used as a security indicator [159], without providing empirical evidence for the relationship between TD and software security. On their part, authors in [160] have advocated the incorporation of misuse or abuse cases into the development process as an independent element. This can potentially help developers vie the software from the attackers' perspective. The relationship between Software Quality Assessment based on Lifecycle Expectations (SQALE) Index and software security risk has been carried out in [161]. The results indicate a statistically significant relationship

between these two constructs. As discussed in [41], developers who care about their users' security and privacy feel encouraged to adopt security practices. In addition, security-conscious software development can be enhanced by the incorporation of security tags within security repositories and user stories. Security tags may be included in the form of S-Tags and S-Marks as highlighted in [24] or as security keywords [162].

The authors in [114] have examined the predictive performance of TD indicators in envisaging software security risks both at project level and at class level of granularity. This was facilitated by building different machine learning (ML) models [163]-[166]. On the other hand, by several experts in the field have identified some close relationship between TD and software security [44], [46]], [45], [48]. In addition, previous researches have attempted to identify software security risk indicators. This involves the ability of software related factors to predict the existence of potential vulnerabilities in software products or components. These factors are identified either from actual vulnerability reports, or through static analysis [90], [147], [91]. Although threat modeling approaches are valuable in software security, they require considerable knowledge of possible technical threats and support from a professional with a detailed understanding of both the industry sector and current cyber threats to it [167]. As such, it may not be a viable solution. In addition, much attention has been given to information retrieved directly from source code, either through static analysis or through text mining. In static analysis, a close correlation may exist between the static analysis alerts density and the actual vulnerabilities that a program contains. This observation is supported by the results of recent empirical studies in [168], [169]. For class-level vulnerability prediction models [170]-[172], Random Forest was has been found to be the best performing ML model [91], [173], [174].

To help software developers understand decision making around security, the authors in [175] have used a facilitated game, *Agile App Security Game* based on the game *Decisions Disruptions*. This game is now utilized extensively in the UK in the management of cyber security training [176]. Static analysis is a software testing mechanism that does not require code execution (unlike dynamic analysis). This enables its early application in the software development [177], [178], [179], [180], [181]. As pointed out in [134], the process of identifying specific kinds of security issues for a given project is an important assurance technique for security. As such, the authors in [8] have proposed DASP, a framework for diagnosing and driving the adoption of software security practices among developers. Table 4 gives a summary of these techniques and their associated weaknesses.

Table 4 Techniques and approaches to software security

Author (s)	Approach/technique	Gap
[32] [33] [61] [86]	Security and Performance (SAP) is an activity that needs to be incorporated in every system software. It enables verification and validation of development process to minimize bugs, errors, mistakes and optimize accuracy and computation cost	SAP verification activities not well planned [200] Lack of awareness on SAP verification activities, should be included in development process. It is not clear how individual security components contribute overall to software security
[87]	The software developers have a perception that writing of code has some elements of security (validation of inputs, use of security mechanisms)	Programmers explicitly depend on conventional security checks such as use of passwords or biometric for login, validation of input fields, abstraction, access modifiers, exception handling These may not be reliable to attacks such as insider threats?
[85]	The author empathizes need to adopt appropriate tools, practices and techniques in order to efficiently manage the process of software development for better results	However the article has not addressed how the process of getting quality tools or techniques that can guaranteebest and trustable secure software
[41] [35]	Encourages every team member (both technical and non-technical) to play a role in security of software from preliminary stages to implementation maintenance and evaluation	More people may gather knowledge of existing vulnerabilities in the software There is no accepted methodology for security in the literature

[75]	Detects denial-of-service attacks and other malwares in blockchain	The challenge with this concept is that it doesn't explicitly address, detection of DoS and malware in other architectures such as Client-Server, federated databases etc.
[89] [90] [91] [103] [104] [105] [106]	software development lifecycle (SDLC) can be used to identify vulnerabilities and risks during development of system it also describes learning models for prediction of vulnerabilities SSDLC complements existing conventional security checks during development Comparison of SSDLC models i.e. System security engineering capability maturity model (SSE-CMM),Microsoft security development lifecycle (MS-SDL) and Software assurance maturity model (SAMM).	Some vulnerabilities may still exists due to integration with other systems which you may not have control Therefore security guru or penetration tester should be included in the development team [194]
[92]	Empirically qualify security tool as a way of managing security in software	Specific to a type of attack (it cannot provide a general solution)
[93] [94] [95]	Emphasis through use of case studies is on the need to align business objectives with security objectives Example; how do you incorporate security in product negotiation?	Some business goals may not be completely mapped to security goals
[96] [97] [98] [99] [100] [101]	The articles discussed ways in which blockchain technology can be used to detect attacks in distributed environment	Need to consider other architectures like client-server and federated environment
[107] [108] [109]	Technical validation should be conducted on the software before its implemented for usage in order to be compliant to security standards	Formal security standards may require certification and continual improvements and updates which might affect overall performance of software
61	Subject the software against risk and threat modeling techniques; interactive application security testing (IAST) Penetration testing; Uses failure-based approach to identify known flaws in software log inspection[110]	There are security biases because the mechanisms, intention and drive of the offenders are diverse
[111] [112]	Experienced based testing can be applied by experts Tools like Arachni, SQL Injection ME, OWASP ZAP, XSS ME, Meta Exploit, NMAP, Burp Suite, and Whatweb can be used by experts acting as malicious attacker in order to identify a flaw in software	Most of these tools do not provide security trust in emerging technologies
[113]	Requires code analysis to be conducted in order to identify violation of coding rules Analysis can be manual of through HP Fortify SCA and Brakeman	Some semantic errors may not be easily detectable such as division by zero that can lead to continuous loop The code rules passed may be independent of process flows Need to create security awareness for developers [127]

[114]	Points out importance of testing software at run-time in order to predict security risks and vulnerabilities early enough	Use of IAST may have security biases inform of data being used for testing
[115], [116] [119] [126]	Articles discuss how trust, dependability and security is achieved through blockchain in order to address data issues	The study is specific to blockchain technology
[117] [118]	Researchers support the argument that security management requires some level of education, self-teaching, osmosis, hands-on learning etc	No empirical evidence to support the argument
[120]	Developed a framework to identify performance deterioration with respect to security	A number of software metrics would need to be considered in determining performance of software security [148], [149], [150], [151]
[130]	Training of developers is encouraged especially on areas they feel comfortable in or areas where security breach is identified	Agile methods or through gamification would be used
[131] [132]	The studies implemented quantum key movement technique to improve network security	Movement of key could require additional overhead
[133]	The study develop DistBlockNet for distributed IoT network architecture in blockchain in order to improve performance in terms of scalability, secure communication and flexibility among devices	No empirical evidence on validity of these technique in security of software over distributed network
[135] [136] [137]	Behavior has some role to play in managing security especially to the one motivating security issues	May not be supported as security method in the literature
[138]	This author [138] argue that the popularity of open-source software could be an indicator of security reliability But results shows that the popularity of open-source is not an indication on security	Need to follow SAP verification activities which are not well planned
[8]	Provides seven considerations that organizations and other parties need to focus on for software security Building security culture Adoption of software security by programmers Understanding risks, trade-off and benefits Provide information for developers to write secure code Tools and process constraints Provide cognitive support for developers Facilitate specific developers security trainings	Lack of this may promote unwanted behavior especially on developers side concerning security issues [140]
[141] [142] [143]	Proposed risk management strategies such as risk identification, risk assessment and analysis	
[38] [144]	According to the author, developing security skills is through sharing of knowledge and talking to each other.	This has not been considered as a tool in security practice

	Peer-based learning [144]	
[145]	Security factors can be assessed and weighted using fuzzy analytic hierarchy process (AHP)	The assessment can only apply in selected software security
[146]	Used logistic regression to predict vulnerability.	Current literature lacks balanced and reliable vulnerability dataset
[90]	Developed models to predict vulnerabilities through text mining	There is no evidence of validation of the predictor against others
[147]	The author alludes to the fact that there is a shift towards deep learning for vulnerability predictors [148], [149], [150], [151] support the idea of having more software metric for vulnerability predictors	However there are certain computation and performance trade-offs

5. Research gaps

The literature reviewed has yielded a number of research gaps that need to be addressed. For instance, although several factors have been studied for their ability to indicate software security risk, very limited attention has been given to TD. This is despite its potential relevance to software security. There is therefore need to investigate the ability of common TD indicators to indicate security risks in software products, both at project level and at class level of granularity. Similarly, the authors in [182] have focused on the DevSecOps framework in a multi-vocal literature survey. However, the attempt made is only to consolidate the definition of the framework rather than identify potential security elements that could be included in the framework or existing secure software development methods.

Numerous and diverse software related factors have been empirically examined for their ability to indicate software security risk. These factors include software metrics [149], text features [90], [147], [91], product popularity [138] and the firm's financial records [34]. However, very limited attention has been given on technical debt. Although agile methods have become popular, they lack complete overview of security issues. There is therefore need for security planning and modeling [184], [184], including security prioritization, monitoring activities and risk management activities. Similarly, vast research endeavours have been directed towards vulnerability prediction [90], [149]; [148], [150], [151], [174]. However, current literature lacks a balanced and reliable vulnerability dataset. Similarly, Building Security In Maturity Model (BSIMM) [185] offers an extensive overview of SSD activities but some roles and artifacts proposed in the literature are not included in this model

Some attempts have been made in defining the concept of Security Debt [44], [46], [45], [48]. In addition, there is a potential shift towards extending the concepts of TD into the security realm. Unfortunately, the literature in the field of software security lacks a well accepted methodology for assessing software security [186], [187], [188], [189]. Similarly, majority of proposed software development approaches remain fully theoretical. This lack of empirical evidence raises concerns about their security enhancing potential, associated costs and suitability for agile methods. To create more effective software security, the association between durability, its attributes, and security for secure SSD needs to be examined.

As pointed out in [190], agile methods have gained popularity and became globally widespread. However, developing secure software with agile methods remains a challenge. As such, gamification elements have been used to incorporate security into agile software development. This has been achieved through the implementation of a variation of Planning Poker as Protection Poker and Threat Poker [191]. On the flip side, motivation is not necessarily achieved through gamification. Although software engineering literature has provided various security elements, there are some key research gaps that hinder the ability to provide secure software. For instance, it is unclear how individual security elements contribute to software security. In addition, it is not clear how these elements impact the agility and costs of software development. The authors in [192] have discussed the existence of various security oriented software development methods following both traditional and agile software development paradigms. The main challenge of these traditional methods is the additional overhead they require, which hinders rapid software development and adaptability to new requirements. As such, it is unclear how secure software development can be executed with agile methods without compromising their agility [193].

Analyzing and reviewing the code for security issues, as well as security testing are vital groups of security elements. Security auditing, code analyses and security tests may be performed continuously within the project cycle. However, different approaches require particular specialized knowledge within the development team. As such, some authors

have proposed the involvement and incorporation of at least one new subject (role) into the existing methodology. In most cases, this is in addition to security guru, security engineers, security masters, security-matter expert and security experts. Therefore, in addition to these roles, a security developer or a penetration tester [194] may be included in the process. The goal of this additional role is to introduce expert knowledge into the development team. This is through the identification of the features that have a security risk based on the security principles, documenting risks in the security backlog, mitigating the identified risks, as well as performing security testing for selected features. Although several approaches are found in literature in which several security elements are proposed, only a fraction of these approaches have been tested in industrial settings. Therefore, little is known about their direct impact on agility and costs. A few of these studies have reported on cost increases and a possible compromise of agility of some elements. However, there is no systematic evaluation of all security elements [195] or their corresponding security element groups regarding their impact on the agility of the software development process and costs of their implementation.

It has been noted that industry practitioners, security tool providers, and researchers have offered standard security guidelines and sophisticated security development tools to ensure a secure software development pipeline. Unfortunately, there continues to be an increase in the number of vulnerabilities that can be exploited by malicious entities. As such, there is need to understand why developers still introduce security vulnerabilities into their applications. In addition, a need arises to understand what can be done to motivate them to write more secure code. According to [196], several technical approaches to prioritization have been developed, some of which prioritize non-functional requirements such as security against functional ones [197]. Unfortunately, there is no evidence in the literature that software product managers have used them in practice. The authors in [75] have also highlighted that the detection and addressing of attacks is an open issue concerning distributed software applications. There is therefore need for organizations to follow the practices that are vital to SAP verification [198]. For enhanced efficiency, these practices must be incorporated into the software development process to identify SAP related defects and avoid failures after deployment. This is only possible when organizations are fully aware of SAP verification activities and appropriately include them in the software development process [199]. However, there is a lack of awareness on the factors that influence SAP verification. This makes it difficult for businesses to improve their verification efforts and ensure that the released software meets these requirements. There is therefore need to identify the mediating factors influencing SAP verification and the actions to promote them. Unfortunately, SAP verification is normally not well planned, and hence the software team must reprioritize verification activities multiple times [200]. This increases the required time and effort. The planning phase includes requirement prioritization, dependency identification, and time and budget for performing SAP verification activities. Therefore, a well-planned and transparent methodology for performing SAP verification is required to smoothly execute the verification process.

6. Conclusion

Software security involves the utilization of various techniques and methods to analyze, defend and mitigate effects of vulnerabilities that may be inherent in software systems. It is therefore important that quality attributes such as reliability, confidentiality, and integrity of software used in these systems be secured. However, it has been shown that software security and durability in the development cycle of software still continue to present new challenges for developers. It has also been revealed that software developers face numerous challenges which hinder their operations. Such problems have been identified as limitations in improvements due to high costs, time-to-market necessities, profitability sway and consumer loyalty concerns. Past research has shown that traditional software development techniques are sequential and rigid. As such, they are not well-suited for the market demands. Although many factors have been studied for their ability to indicate software security risk, very limited attention has been given to issues such as technical debt. The literature has shown that great research endeavours have been directed towards vulnerability prediction. However, there is lack of some balanced and reliable vulnerability datasets. Some authors have pointed out that agile methods have gained popularity and became globally widespread during software development. However, developing secure software with agile methods remains a challenge. Future research work will involve the development of novel mechanisms of addressing some of these software security issues during the SDLC procedures.

Compliance with ethical standards

Acknowledgments

The authors would like to extend their gratitude to all colleagues who offered some support during the development of this article.

Disclosure of conflict of interest

The authors declare that they have no any conflict of interest.

References

[1] Mirakhorli M, Galster M, Williams L. Understanding software security from design to deployment. ACM SIGSOFT Software Engineering Notes. 2020 Apr 30, 45(2):25-6.

[2] Zhang W, Ding DS, Sheng YB, Zhou L, Shi BS, Guo GC. Quantum secure direct communication with quantum memory. Physical review letters. 2017 May 31, 118(22):220501.

[3] Kasauli R, Knauss E, Kanagwa B, Nilsson A, Calikli G. Safety-critical systems and agile development: A mapping study. In2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) 2018 Aug 29 (pp. 470-477). IEEE.

[4] Alyami H, Nadeem M, Alharbi A, Alosaimi W, Ansari MT, Pandey D, Kumar R, Khan RA. The evaluation of software security through quantum computing techniques: A durability perspective. Applied Sciences. 2021 Dec 11, 11(24):11784.

[5] Nyangaresi VO. Privacy Preserving Three-factor Authentication Protocol for Secure Message Forwarding in Wireless Body Area Networks. Ad Hoc Networks. 2023 Apr 1, 142:103117.

[6] Pang XL, Qiao LF, Sun K, Liu Y, Yang AL, Jin XM. Experimental quantum-enhanced cryptographic remote control. Scientific Reports. 2019 Apr 9, 9(1):1-5.

[7] Arute F, Arya K, Babbush R, Bacon D, Bardin JC, Barends R, Biswas R, Boixo S, Brandao FG, Buell DA, Burkett B. Quantum supremacy using a programmable superconducting processor. Nature. 2019 Oct, 574(7779):505-10.

[8] Larios-Vargas E, Elazhary O, Yousefi S, Lowlind D, Vliek ML, Storey MA. DASP: A Framework for Driving the Adoption of Software Security Practices. IEEE Transactions on Software Engineering. 2023 Jan 10.

[9] Sen J, editor. Theory and practice of cryptography and network security protocols and technologies. BoD–Books on Demand, 2013 Jul 17.

[10] Bernstein DJ, Lange T. Post-quantum cryptography. Nature. 2017 Sep 14, 549(7671):188-94.

[11] Alashaikh A, Tipper D, Gomes T. Embedded network design to support availability differentiation. Annals of Telecommunications. 2019 Oct, 74:605-23.

[12] Nyangaresi VO, Ogundoyin SO. Certificate based authentication scheme for smart homes. In2021 3rd Global Power, Energy and Communication Conference (GPECOM) 2021 Oct 5 (pp. 202-207). IEEE.

[13] Alzahrani FA, Ahmad M, Nadeem M, Kumar R, Khan RA. Integrity assessment of medical devices for improving hospital services. Comput. Mater. Contin. 2021 Jan 1, 67(3).

[14] Agrawal A, Alenezi M, Khan SA, Kumar R, Khan RA. Multi-level fuzzy system for usable-security assessment. Journal of King Saud University-Computer and Information Sciences. 2022 Mar 1, 34(3):657-65.

[15] Saeedi K, Visvizi A. Software development methodologies, HEIs, and the digital economy. Education Sciences. 2021 Feb 13, 11(2):73.

[16] Mihelič A, Hovelja T, Vrhovec SL. Towards a delegation-type secure software development method. InProceedings of the Third Central European Cybersecurity Conference 2019 Nov 14 (pp. 1-2).

[17] Rashed AN, Ahammad SH, Daher MG, Sorathiya V, Siddique A, Asaduzzaman S, Rehana H, Dutta N, Patel SK, Nyangaresi VO, Jibon RH. Spatial single mode laser source interaction with measured pulse based parabolic index multimode fiber. Journal of Optical Communications. 2022 Jun 21.

[18] Nowroozi A, Teymoori P, Ramezanifarkhani T, Besharati MR, Izadi M. A crisis situations decision-making systems software development process with rescue experiences. IEEE Access. 2020 Mar 19, 8:59599-617.

[19] Hering D, Schwartz T, Boden A, Wulf V. Integrating usability-engineering into the software developing processes of SME: A case study of software developing SME in Germany. In2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering 2015 May 18 (pp. 121-122). IEEE.

[20] Oueslati H, Rahman MM, ben Othmane L. Literature review of the challenges of developing secure software using the agile approach. In2015 10th International Conference on Availability, Reliability and Security 2015 Aug 24 (pp. 540-547). IEEE.

[21] Rindell K, Ruohonen J, Holvitie J, Hyrynsalmi S, Leppänen V. Security in agile software development: A practitioner survey. Information and Software Technology. 2021 Mar 1, 131:106488.

[22] ZakiRashed AN, Ahammad SH, Daher MG, Sorathiya V, Siddique A, Asaduzzaman S, Rehana H, Dutta N, Patel SK, Nyangaresi VO, Jibon RH. Signal propagation parameters estimation through designed multi layerfibre with higher dominant modes using OptiFibre simulation. Journal of Optical Communications. 2022 Jun 23(0).

[23] Adelyar SH, Norta A. Towards a secure agile software development process. In2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC) 2016 Sep 6 (pp. 101-106). IEEE.

[24] Pohl C, Hof HJ. Secure scrum: Development of secure software with scrum. arXiv preprint arXiv:1507.02992. 2015 Jul 10.

[25] Abrahamsson P, Oza N, Siponen MT. Agile Software Development Method, A Comparative Review1. arXiv preprint arXiv:1903.10913. 2019 Mar 26.

[26] Cobb CG. The project manager's guide to mastering Agile: Principles and practices for an adaptive approach. John Wiley & Sons, 2023 Apr 4.

[27] Eid MM, Arunachalam R, Sorathiya V, Lavadiya S, Patel SK, Parmar J, Delwar TS, Ryu JY, Nyangaresi VO, Rashed AN. QAM receiver based on light amplifiers measured with effective role of optical coherent duobinary transmitter. Journal of Optical Communications. 2022 Jan 17.

[28] Cico O, Jaccheri L, Nguyen-Duc A, Zhang H. Exploring the intersection between software industry and Software Engineering education-A systematic mapping of Software Engineering Trends. Journal of Systems and Software. 2021 Feb 1, 172:110736.

[29] Alenezi M, Kumar R, Agrawal A, Khan RA. Usable-security attribute evaluation using fuzzy analytic hierarchy process. ICIC Express Lett. An Int. J. Res. Surv. 2019, 13(6).

[30] Votipka D, Fulton KR, Parker J, Hou M, Mazurek ML, Hicks M. Understanding security mistakes developers make: Qualitative analysis from build it, break it, fix it. In29th USENIX Security Symposium (USENIX Security 20) 2020 (pp. 109-126).

[31] Weir C, Becker I, Blair L. Incorporating software security: using developer workshops to engage product managers. Empirical Software Engineering. 2023 Mar, 28(2):21.

[32] Ribeiro VV, Cruzes DS, Travassos GH. Moderator factors of software security and performance verification. Journal of Systems and Software. 2022 Feb 1, 184:111137.

[33] Nyangaresi VO. Lightweight anonymous authentication protocol for resource-constrained smart home devices based on elliptic curve cryptography. Journal of Systems Architecture. 2022 Dec 1, 133:102763.

[34] Roumani Y, Nwankpa JK, Roumani YF. Examining the relationship between firm's financial records and security vulnerabilities. International Journal of Information Management. 2016 Dec 1, 36(6):987-94.

[35] Assal H, Chiasson S. Motivations and amotivations for software security. InSOUPS Workshop on Security Information Workers (WSIW). USENIX Association 2018 Aug (pp. 1-4).

[36] Yuce B, Schaumont P, Witteman M. Fault attacks on secure embedded software: Threats, design, and evaluation. Journal of Hardware and Systems Security. 2018 Jun, 2:111-30.

[37] Weir C, Becker I, Noble J, Blair L, Sasse MA, Rashid A. Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers. Software: Practice and Experience. 2020 Mar, 50(3):275-98.

[38] Sajwan L, Noble J, Anslow C, Biddle R. Why do programmers do what they do? a theory of influences on security practices. InWorkshop on Usable Security and Privacy 2021 May.

[39] Mohammad Z, Nyangaresi V, Abusukhon A. On the Security of the Standardized MQV Protocol and Its Based Evolution Protocols. In2021 International Conference on Information Technology (ICIT) 2021 Jul 14 (pp. 320-325). IEEE.

[40] Song M, Wang P, Yang P. Promotion of secure software development assimilation: stimulating individual motivation. Chinese Management Studies. 2018 Feb 2.

[41] Assal H, Chiasson S. 'Think secure from the beginning' A Survey with Software Developers. InProceedings of the 2019 CHI conference on human factors in computing systems 2019 May 2 (pp. 1-13).

[42] Poller A, Kocksch L, Türpe S, Epp FA, Kinder-Kurlanda K. Can security become a routine? A study of organizational change in an agile software development group. InProceedings of the 2017 ACM conference on computer supported cooperative work and social computing 2017 Feb 25 (pp. 2489-2503).

[43] Hussain MA, Hussien ZA, Abduljabbar ZA, Ma J, Al Sibahee MA, Hussain SA, Nyangaresi VO, Jiao X. Provably throttling SQLI using an enciphering query and secure matching. Egyptian Informatics Journal. 2022 Dec 1, 23(4):145-62.

[44] Rindell K, Bernsmed K, Jaatun MG. Managing security in software: Or: How i learned to stop worrying and manage the security technical debt. InProceedings of the 14th International Conference on Availability, Reliability and Security 2019 Aug 26 (pp. 1-8).

[45] Izurieta C, Rice D, Kimball K, Valentien T. A position study to investigate technical debt associated with security weaknesses. InProceedings of the 2018 International Conference on technical debt 2018 May 27 (pp. 138-142).

[46] Rindell K, Holvitie J. Security risk assessment and management as technical debt. In2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security) 2019 Jun 3 (pp. 1-8). IEEE.

[47] Abduljaleel IQ, Abduljabbar ZA, Al Sibahee MA, Ghrabat MJ, Ma J, Nyangaresi VO. A Lightweight Hybrid Scheme for Hiding Text Messages in Colour Images Using LSB, Lah Transform and Chaotic Techniques. Journal of Sensor and Actuator Networks. 2022 Dec, 11(4):66.

[48] Izurieta C, Prouty M. Leveraging secdevops to tackle the technical debt associated with cybersecurity attack tactics. In2019 IEEE/ACM International Conference on Technical Debt (TechDebt) 2019 May 26 (pp. 33-37). IEEE.

[49] Rauf I, Petre M, Tun T, Lopez T, Lunn P, Van der Linden D, Towse J, Sharp H, Levine M, Rashid A, Nuseibeh B. The case for adaptive security interventions. ACM Transactions on Software Engineering and Methodology (TOSEM). 2021 Sep 28, 31(1):1-52.

[50] Mokhberi A, Beznosov K. SoK: human, organizational, and technological dimensions of developers' challenges in engineering secure software. InProceedings of the 2021 European Symposium on Usable Security 2021 Oct 11 (pp. 59-75).

[51] Poth A, Kottke M, Middelhauve K, Mahr T, Riel A. Lean integration of IT security and data privacy governance aspects into product development in agile organizations. J. Univers. Comput. Sci.. 2021 Jan 1, 27(8):868-93.

[52] Soualmi A, Laouamer L, Alti A. Performing Security on Digital Images. InExploring Security in Software Architecture and Design 2019 (pp. 211-246). IGI Global.

[53] Nyangaresi VO. Provably Secure Pseudonyms based Authentication Protocol for Wearable Ubiquitous Computing Environment. In2022 International Conference on Inventive Computation Technologies (ICICT) 2022 Jul 20 (pp. 1-6). IEEE.

[54] Tøndel IA, Jaatun MG. Towards a conceptual framework for security requirements work in agile software development. InResearch Anthology on Agile Software, Software Development, and Testing 2022 (pp. 247-279). IGI Global.

[55] Türpe S, Poller A. Managing Security Work in Scrum: Tensions and Challenges. SecSE@ ESORICS. 2017 Jan, 2017:34-49.

[56] Ansari MT, Al-Zahrani FA, Pandey D, Agrawal A. A fuzzy TOPSIS based analysis toward selection of effective security requirements engineering approach for trustworthy healthcare software development. BMC Medical Informatics and Decision Making. 2020 Dec, 20(1):1-3.

[57] Aljaž T. Improving throughput and due date performance of IT DevOps teams. ElektrotehniskiVestnik. 2021 May 30, 88(3):121-8.

[58] Alsamhi SH, Shvetsov AV, Kumar S, Shvetsova SV, Alhartomi MA, Hawbani A, Rajput NS, Srivastava S, Saif A, Nyangaresi VO. UAV computing-assisted search and rescue mission framework for disaster and harsh environment mitigation. Drones. 2022 Jun 22, 6(7):154.

[59] Braz L, Aeberhard C, Çalikli G, Bacchelli A. Less is more: supporting developers in vulnerability detection during code review. InProceedings of the 44th International Conference on Software Engineering 2022 May 21 (pp. 1317-1329).

[60] Humayun M, Jhanjhi N, Almufareh MF, Khalil MI. Security Threat and Vulnerability Assessment and Measurement in Secure Software Development. Comput. Mater. Contin. 2022 Jan 1, 71:5039-59.

[61] Almufareh MF, Humayun M. Improving the Safety and Security of Software Systems by Mediating SAP Verification. Applied Sciences. 2023 Jan, 13(1):647.

[62] Yarza I, Agirre I, Mugarza I, Cerrolaza JP. Safety and security collaborative analysis framework for high-performance embedded computing devices. Microprocessors and Microsystems. 2022 Sep 1, 93:104572.

[63] Nyangaresi VO. Masked Symmetric Key Encrypted Verification Codes for Secure Authentication in Smart Grid Networks. In 2022 4th Global Power, Energy and Communication Conference (GPECOM) 2022 Jun 14 (pp. 427-432).

[64] Luszcz J. Apache struts 2: how technical and development gaps caused the equifax breach. Network Security. 2018 Jan 1, 2018(1):5-8.

[65] Liang X, Shetty S, Tosh D, Kamhoua C, Kwiat K, Njilla L. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) 2017 May 14 (pp. 468-477). IEEE.

[66] Arbain AF, Jawawi DN, Kadir WM, Ghani I. Case study on non-functional requirement change impact traceability for Agile software development. International Journal on Advanced Science, Engineering and Information Technology. 2020(1):34-40.

[67] Rahman MS, Reza H. Systematic mapping study of non-functional requirements in big data system. In2020 IEEE International Conference on Electro Information Technology (EIT) 2020 Jul 31 (pp. 025-031). IEEE.

[68] Nyakomitta SP, Omollo V. Biometric-Based Authentication Model for E-Card Payment Technology. IOSR Journal of Computer Engineering (IOSRJCE). 2014, 16(5):137-44.

[69] Alwadi A, Nahhas A, Bosse S, Jamous N, Turowski K. A Modernized Model for Performance Requirements and their Interdependencies. In2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA) 2019 Nov 3 (pp. 1-8). IEEE.

[70] Kosti MV, Ampatzoglou A, Chatzigeorgiou A, Pallas G, Stamelos I, Angelis L. Technical debt principal assessment through structural metrics. In2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA) 2017 Aug 30 (pp. 329-333). IEEE.

[71] Tsoukalas D, Kehagias D, Siavvas M, Chatzigeorgiou A. Technical debt forecasting: an empirical study on open-source repositories. Journal of Systems and Software. 2020 Dec 1, 170:110777.

[72] Alves NS, Mendes TS, de Mendonça MG, Spínola RO, Shull F, Seaman C. Identification and management of technical debt: A systematic mapping study. Information and Software Technology. 2016 Feb 1, 70:100-21.

[73] Siebra CA, Cavalcanti A, Silva FQ, Santos AL, Gouveia TB. Applying metrics to identify and monitor technical debt items during software evolution. In2014 IEEE International Symposium on Software Reliability Engineering Workshops 2014 Nov 3 (pp. 92-95). IEEE.

[74] Nyangaresi VO, Ma J. A Formally Verified Message Validation Protocol for Intelligent IoT E-Health Systems. In2022 IEEE World Conference on Applied Intelligence and Computing (AIC) 2022 Jun 17 (pp. 416-422). IEEE.

[75] Algarni A, Attaallah A, Eassa F, Khemakhem M, Jambi K, Aljihani H, Almarhabi K, Albalwy F. A security testing mechanism for detecting attacks in distributed software applications using blockchain. PloS one. 2023 Jan 20, 18(1):e0280038.

[76] Saca MA. Refactoring improving the design of existing code. In2017 IEEE 37th Central America and Panama Convention (CONCAPAN XXXVII) 2017 Nov 15 (pp. 1-3). IEEE.

[77] Palomba F, Bavota G, Di Penta M, Fasano F, Oliveto R, De Lucia A. On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. InProceedings of the 40th International Conference on Software Engineering 2018 May 27 (pp. 482-482).

[78] Xuan J, Hu Y, Jiang H. Debt-prone bugs: technical debt in software maintenance. arXiv preprint arXiv:1704.04766. 2017 Apr 16.

[79] Griffith I, Reimanis D, Izurieta C, Codabux Z, Deo A, Williams B. The correspondence between software quality models and technical debt estimation approaches. In2014 sixth international workshop on managing technical debt 2014 Sep 30 (pp. 19-26). IEEE.

[80] Digkas G, Lungu M, Chatzigeorgiou A, Avgeriou P. The evolution of technical debt in the apache ecosystem. InSoftware Architecture: 11th European Conference, ECSA 2017, Canterbury, UK, September 11-15, 2017, Proceedings 11 2017 (pp. 51-66). Springer International Publishing.

[81] Al-Wesabi FN. Improving Availability in Component-Based Distributed Systems. Intelligent Automation & Soft Computing. 2020 Nov 1, 26(6).

[82] Ali M, Bagchi S. Hybrid Architecture for Autonomous Load Balancing in Distributed Systems based on Smooth Fuzzy Function. Intelligent Automation & Soft Computing. 2018 Dec 1, 24(4).

[83] Nyangaresi VO. A Formally Validated Authentication Algorithm for Secure Message Forwarding in Smart Home Networks. SN Computer Science. 2022 Jul 9, 3(5):364.

[84] Oyetoyan TD, Milosheska B, Grini M, Soares Cruzes D. Myths and facts about static application security testing tools: an action research at Telenor digital. InAgile Processes in Software Engineering and Extreme Programming: 19th International Conference, XP 2018, Porto, Portugal, May 21–25, 2018, Proceedings 19 2018 (pp. 86-103). Springer International Publishing.

[85] Bianchi MJ, Conforto EC, Amaral DC. Beyond the agile methods: A diagnostic tool to support the development of hybrid models. International Journal of Managing Projects in Business. 2021 Feb 18.

[86] Aruna ER, Rama Mohan Reddy A, Sunitha KV. Secure SDLC Using Security Patterns 2.0. InIOT with Smart Systems: Proceedings of ICTIS 2021, Volume 2 2022 (pp. 699-708). Springer Singapore.

[87] Van Der Linden D, Anthonysamy P, Nuseibeh B, Tun TT, Petre M, Levine M, Towse J, Rashid A. Schrödinger's security: opening the box on app developers' security rationale. InProceedings of the ACM/IEEE 42nd International Conference on Software Engineering 2020 Jun 27 (pp. 149-160).

[88] Abood EW, Abdullah AM, Al Sibahe MA, Abduljabbar ZA, Nyangaresi VO, Kalafy SA, Ghrabta MJ. Audio steganography with enhanced LSB method for securing encrypted text with bit cycling. Bulletin of Electrical Engineering and Informatics. 2022 Feb 1, 11(1):185-94.

[89] Siavvas M, Gelenbe E, Kehagias D, Tzovaras D. Static analysis-based approaches for secure software development. InSecurity in Computer and Information Sciences: First International ISCIS Security Workshop 2018, Euro-CYBERSEC 2018, London, UK, February 26-27, 2018, Revised Selected Papers 1 2018 (pp. 142-157). Springer International Publishing.

[90] Scandariato R, Walden J, Hovsepyan A, Joosen W. Predicting vulnerable software components via text mining. IEEE Transactions on Software Engineering. 2014 Jul 18, 40(10):993-1006.

[91] Dam HK, Tran T, Pham T, Ng SW, Grundy J, Ghose A. Automatic feature learning for predicting vulnerable software components. IEEE Transactions on Software Engineering. 2018 Nov 18, 47(1):67-85.

[92] Parizi RM, Dehghantanha A, Choo KK, Singh A. Empirical vulnerability analysis of automated smart contracts security testing on blockchains. arXiv preprint arXiv:1809.02702. 2018 Sep 7.

[93] Caputo DD, Pfleeger SL, Sasse MA, Ammann P, Offutt J, Deng L. Barriers to usable security? Three organizational case studies. IEEE Security & Privacy. 2016 Oct 25, 14(5):22-32.

[94] Nyangaresi VO, Ahmad M, Alkhayyat A, Feng W. Artificial neural network and symmetric key cryptography based verification protocol for 5G enabled Internet of Things. Expert Systems. 2022 Dec, 39(10):e13126.

[95] Weir C, Rashid A, Noble J. Challenging software developers: dialectic as a foundation for security assurance techniques. Journal of Cybersecurity. 2020, 6(1):tyaa007.

[96] Rathore S, Park JH. A blockchain-based deep learning approach for cyber security in next generation industrial cyber-physical systems. IEEE Transactions on Industrial Informatics. 2020 Nov 26, 17(8):5522-32.

[97] Wen Y, Lu F, Liu Y, Huang X. Attacks and countermeasures on blockchains: A survey from layering perspective. Computer Networks. 2021 May 22, 191:107978.

[98] Saveetha D, Maragatham G. Design of Blockchain enabled intrusion detection model for detecting security attacks using deep learning. Pattern Recognition Letters. 2022 Jan 1, 153:24-8.

[99] Rathore S, Park JH, Chang H. Deep learning and blockchain-empowered security framework for intelligent 5G-enabled IoT. IEEE Access. 2021 May 3, 9:90075-83.

[100] Sharma PK, Kumar N, Park JH. Blockchain technology toward green IoT: Opportunities and challenges. IEEE Network. 2020 Mar 3, 34(4):263-9.

[101] Singh SK, Azzaoui AE, Kim TW, Pan Y, Park JH. DeepBlockScheme: A deep learning-based blockchain driven scheme for secure smart city. Human-centric Computing and Information Sciences. 2021 Mar 15, 11(12):1-3.

[102] Nyangaresi VO. ECC based authentication scheme for smart homes. In2021 International Symposium ELMAR 2021 Sep 13 (pp. 5-10). IEEE.

[103] Khan RA, Khan SU, Khan HU, Ilyas M. Systematic literature review on security risks and its practices in secure software development. ieee Access. 2022 Jan 5, 10:5456-81.

[104] Rodriguez M, Piattini M, Ebert C. Software verification and validation technologies and tools. IEEE Software. 2019 Feb 21, 36(2):13-24.

[105] Fujdiak R, Mlynek P, Mrnustik P, Barabas M, Blazek P, Borcik F, Misurec J. Managing the secure software development. In2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS) 2019 Jun 24 (pp. 1-4). IEEE.

[106] Kamal AH, Yen CC, Hui GJ, Ling PS. Risk Assessment, Threat Modeling and Security Testing in SDLC. arXiv preprint arXiv:2012.07226. 2020 Dec 14.

[107] Ribeiro VV, Cruzes DS, Travassos GH. A perception of the practice of software security and performance verification. In2018 25th Australasian Software Engineering Conference (ASWEC) 2018 Nov 26 (pp. 71-80). IEEE.

[108] Varela-Vaca ÁJ, Rosado DG, Sánchez LE, Gómez-López MT, Gasca RM, Fernández-Medina E. CARMEN: A framework for the verification and diagnosis of the specification of security requirements in cyber-physical systems. Computers in Industry. 2021 Nov 1, 132:103524.

[109] Nyangaresi VO. Provably secure protocol for 5G HetNets. In2021 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS) 2021 Nov 1 (pp. 17-22). IEEE.

[110] Zhu J, He S, Liu J, He P, Xie Q, Zheng Z, Lyu MR. Tools and benchmarks for automated log parsing. In2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) 2019 May 25 (pp. 121-130). IEEE.

[111] Khan S, Parkinson S. Discovering and utilising expert knowledge from security event logs. Journal of Information Security and Applications. 2019 Oct 1, 48:102375.

[112] Dieber B, White R, Taurer S, Breiling B, Caiazza G, Christensen H, Cortesi A. Penetration testing ROS. Robot Operating System (ROS) The Complete Reference (Volume 4). 2020:183-225.

[113] Hong K. Performance, Security, and Safety Requirements Testing for Smart Systems Through Systematic Software Analysis (Doctoral dissertation).

[114] Siavvas M, Tsoukalas D, Jankovic M, Kehagias D, Tzovaras D. Technical debt as an indicator of software security risk: a machine learning approach for software development enterprises. Enterprise Information Systems. 2022 May 4, 16(5):1824017.

[115] Smith TD. The blockchain litmus test. In2017 IEEE International Conference on Big Data (Big Data) 2017 Dec 11 (pp. 2299-2308). IEEE.

[116] Nyangaresi VO. Terminal independent security token derivation scheme for ultra-dense IoT networks. Array. 2022 Sep 1, 15:100210.

[117] Weir C, Becker I, Blair L. A passion for security: Intervening to help software developers. In2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) 2021 May 25 (pp. 21-30). IEEE.

[118] Yskout K, Scandariato R, Joosen W. Do security patterns really help designers?. In2015 IEEE/ACM 37th IEEE International Conference on Software Engineering 2015 May 16 (Vol. 1, pp. 292-302). IEEE.

[119] Roy S, Ashaduzzaman M, Hassan M, Chowdhury AR. Blockchain for IoT security and management: Current prospects, challenges and future directions. In2018 5th International Conference on Networking, Systems and Security (NSysS) 2018 Dec 18 (pp. 1-9). IEEE.

[120] Cortellessa V, Trubiani C, Mostarda L, Dulay N. An architectural framework for analyzing tradeoffs between software security and performance. InArchitecting Critical Systems: First International Symposium, ISARCS 2010, Prague, Czech Republic, June 23-25, 2010 Proceedings 2010 (pp. 1-18). Springer Berlin Heidelberg.

[121] Rindell K, Hyrynsalmi S, Leppänen V. Securing Scrum for VAHTI. InSPLST 2015 Jun (pp. 236-250).

[122] benOthmane L, Angin P, Weffers H, Bhargava B. Extending the agile development process to develop acceptably secure software. IEEE Transactions on dependable and secure computing. 2014 Jan 9, 11(6):497-509.

[123] Nyangaresi VO. Hardware assisted protocol for attacks prevention in ad hoc networks. InEmerging Technologies in Computing: 4th EAI/IAER International Conference, iCETiC 2021, Virtual Event, August 18–19, 2021, Proceedings 4 2021 (pp. 3-20). Springer International Publishing.

[124] Giacalone M, Paci F, Mammoliti R, Perugino R, Massacci F, Selli C. Security triage: an industrial case study on the effectiveness of a lean methodology to identify security requirements. InProceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement 2014 Sep 18 (pp. 1-8).

[125] Maria RE, Rodrigues Jr LA, Pinto NA. ScrumS: a model for safe agile development. InProceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems 2015 Oct 25 (pp. 43-47).

[126] Tøndel IA, Cruzes DS, Jaatun MG, Rindell K. The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects. InProceedings of the 14th International Conference on Availability, Reliability and Security 2019 Aug 26 (pp. 1-8).

[127] Lopez T, Sharp H, Tun T, Bandara A, Levine M, Nuseibeh B. " Hopefully We Are Mostly Secure": Views on Secure Code in Professional Practice. In2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE) 2019 May 27 (pp. 61-68). IEEE.

[128] Lopez T, Sharp H, Tun T, Bandara A, Levine M, Nuseibeh B. Talking about security with professional developers. In2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP) 2019 May 28 (pp. 34-40). IEEE.

[129] Beyer M, Ahmed S, Doerlemann K, Arnell S, Parkin S, Sasse A, Passingham N. Awareness is only the first step: A framework for progressive engagement of staff in cyber security. Hewlett Packard, Business white paper. 2016.

[130] Nyangaresi VO. Lightweight key agreement and authentication protocol for smart homes. In2021 IEEE AFRICON 2021 Sep 13 (pp. 1-6). IEEE.

[131] Mitra S, Jana B, Bhattacharya S, Pal P, Poray J. Quantum cryptography: Overview, security issues and future challenges. In2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix) 2017 Nov 2 (pp. 1-7). IEEE.

[132] Abomhara M, Køien GM. Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. Journal of Cyber Security and Mobility. 2015 May 22:65-88.

[133] Sharma PK, Singh S, Jeong YS, Park JH. Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks. IEEE Communications Magazine. 2017 Sep 8, 55(9):78-85.

[134] Such JM, Gouglidis A, Knowles W, Misra G, Rashid A. Information assurance techniques: Perceived cost effectiveness. Computers & Security. 2016 Jul 1, 60:117-33.

[135] Haney J, Lutters W, Jacobs J. Cybersecurity Advocates: Force Multipliers in Security Behavior Change. IEEE Security & Privacy. 2021 Jul 5, 19(4):54-9.

[136] Pfleeger SL, Sasse MA, Furnham A. From weakest link to security hero: Transforming staff security behavior. Journal of Homeland Security and Emergency Management. 2014 Dec 1, 11(4):489-510.

[137] Nyangaresi VO, Petrovic N. Efficient PUF based authentication protocol for internet of drones. In2021 International Telecommunications Conference (ITC-Egypt) 2021 Jul 13 (pp. 1-4). IEEE.

[138] Siavvas M, Jankovic M, Kehagias D, Tzovaras D. Is popularity an indicator of software security?. In2018 International Conference on intelligent systems (IS) 2018 Sep 25 (pp. 692-697). IEEE.

[139] Bell L, Brunton-Spall M, Smith R, Bird J. Agile application security: enabling security in a continuous delivery pipeline. " O'Reilly Media, Inc.", 2017 Sep 8.

[140] Becker I, Parkin S, Sasse MA. Finding security champions in blends of organisational culture. Proc. USEC. 2017, 11.

[141] Maier P, Ma Z, Bloem R. Towards a secure scrum process for agile web application development. InProceedings of the 12th International Conference on Availability, Reliability and Security 2017 Aug 29 (pp. 1-8).

[142] Baca D, Boldt M, Carlsson B, Jacobsson A. A novel security-enhanced agile software development process applied in an industrial setting. In2015 10th International Conference on Availability, Reliability and Security 2015 Aug 24 (pp. 11-19). IEEE.

[143] Stålhane T, Johnsen SO. Resilience and safety in agile development (Through safescrum). Safety and Reliability–Theory and Applications. 2017.

[144] Nyangaresi VO, Mohammad Z. Privacy preservation protocol for smart grid networks. In2021 International Telecommunications Conference (ITC-Egypt) 2021 Jul 13 (pp. 1-4). IEEE.

[145] Ma X, Zeng P, Zhou H. Phase-matching quantum key distribution. Physical Review X. 2018 Aug 16, 8(3):031043.

[146] Camilo F, Meneely A, Nagappan M. Do bugs foreshadow vulnerabilities? a study of the chromium project. In2015 IEEE/ACM 12th Working Conference on Mining Software Repositories 2015 May 16 (pp. 269-279). IEEE.

[147] Pang Y, Xue X, Wang H. Predicting vulnerable software components through deep neural network. InProceedings of the 2017 International Conference on Deep Learning Technologies 2017 Jun 2 (pp. 6-10).

[148] Moshtari S, Sami A. Evaluating and comparing complexity, coupling and a new proposed set of coupling metrics in cross-project vulnerability prediction. InProceedings of the 31st Annual ACM Symposium on Applied Computing 2016 Apr 4 (pp. 1415-1421).

[149] Siavvas M, Kehagias D, Tzovaras D. A preliminary study on the relationship among software metrics and specific vulnerability types. In2017 International Conference on Computational Science and Computational Intelligence (CSCI) 2017 Dec 14 (pp. 916-921). IEEE.

[150] Ferenc R, Hegedűs P, Gyimesi P, Antal G, Bán D, Gyimóthy T. Challenging machine learning algorithms in predicting vulnerable javascript functions. In2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE) 2019 May 28 (pp. 8-14). IEEE.

[151] Jimenez M, Rwemalika R, Papadakis M, Sarro F, Le Traon Y, Harman M. The importance of accounting for real-world labelling when predicting software vulnerabilities. InProceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering 2019 Aug 12 (pp. 695-705).

[152] Nyangaresi VO, Moundounga AR. Secure data exchange scheme for smart grids. In2021 IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI) 2021 Sep 6 (pp. 312-316). IEEE.

[153] Ahola J, Frühwirth C, Helenius M, Kutvonen L, Myllylahti J, Nyberg T, Ari P, Pietikäinen P, Röning J, Ruohomaa S, Särs C. Handbook of the secure agile software development life cycle.

[154] Vidoni M, Codabux Z, Fard FH. Infinite technical debt. Journal of Systems and Software. 2022 Apr 22:111336.

[155] Borup NB, Christiansen AL, Tovgaard SH, Persson JS. Deliberative technical debt management: An action research study. InSoftware Business: 12th International Conference, ICSOB 2021, Drammen, Norway, December 2–3, 2021, Proceedings 12 2021 (pp. 50-65). Springer International Publishing.

[156] Verdecchia R, Malavolta I, Lago P, Ozkaya I. Empirical evaluation of an architectural technical debt index in the context of the Apache and ONAP ecosystems. PeerJ Computer Science. 2022 Feb 7, 8:e833.

[157] Letouzey JL. Managing large application portfolio with technical debt related measures. In2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA) 2016 Oct 1 (pp. 181-181). IEEE Computer Society.

[158] Giraldo FD, España S, Pineda MA, Giraldo WJ, Pastor O. Conciliating model-driven engineering with technical debt using a quality framework. InInformation Systems Engineering in Complex Environments: CAiSE Forum 2014, Thessaloniki, Greece, June 16-20, 2014, Selected Extended Papers 26 2015 (pp. 199-214). Springer International Publishing.

[159] Nyangaresi VO, Morsy MA. Towards privacy preservation in internet of drones. In2021 IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI) 2021 Sep 6 (pp. 306-311). IEEE.

[160] Lee KH, Park YB. Adaption of integrated secure guide for secure software development lifecycle. International Journal of Security and Its Applications. 2016 Jun, 10(6):145-54.

[161] Siavvas M, Tsoukalas D, Jankovic M, Kehagias D, Chatzigeorgiou A, Tzovaras D, Anicic N, Gelenbe E. An empirical evaluation of the relationship between technical debt and software security. In9th International Conference on Information society and technology (ICIST) 2019 (Vol. 2019).

[162] Koç G, Aydos M, Tekerek M. Evaluation of trustworthy scrum employment for agile software development based on the views of software developers. In2019 4th International Conference on Computer Science and Engineering (UBMK) 2019 Sep 11 (pp. 63-67). IEEE.

[163] Bannigan P, Bao Z, Hickman RJ, Aldeghi M, Häse F, Aspuru-Guzik A, Allen C. Machine learning models to accelerate the design of polymeric long-acting injectables. Nature Communications. 2023 Jan 10, 14(1):35.

[164] Huyut MT. Automatic detection of severely and mildly infected COVID-19 patients with supervised machine learning models. IRBM. 2023 Feb 1, 44(1):100725.

[165] Dritsas E, Trigka M. Supervised Machine Learning Models for Liver Disease Risk Prediction. Computers. 2023 Jan 13, 12(1):19.

[166] Nyangaresi VO, El-Omari NK, Nyakina JN. Efficient Feature Selection and ML Algorithm for Accurate Diagnostics. Journal of Computer Science Research. 2022 Jan 25, 4(1):10-9.

[167] Shostack A. Threat modeling: Designing for security. John Wiley & Sons, 2014 Feb 12.

[168] Walden J, Stuckman J, Scandariato R. Predicting vulnerable components: Software metrics vs text mining. In2014 IEEE 25th international symposium on software reliability engineering 2014 Nov 3 (pp. 23-33). IEEE.

[169] Yang J, Ryu D, Baik J. Improving vulnerability prediction accuracy with secure coding standard violation measures. In2016 International Conference on Big Data and Smart Computing (BigComp) 2016 Jan 18 (pp. 115-122). IEEE.

[170] Stiglic G, Kocbek P, Fijacko N, Zitnik M, Verbert K, Cilar L. Interpretability of machine learning-based prediction models in healthcare. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2020 Sep, 10(5):e1379.

[171] Collins GS, Moons KG. Reporting of artificial intelligence prediction models. The Lancet. 2019 Apr 20, 393(10181):1577-9.

[172] Al Sibahee MA, Ma J, Nyangaresi VO, Abduljabbar ZA. Efficient Extreme Gradient Boosting Based Algorithm for QoS Optimization in Inter-Radio Access Technology Handoffs. In2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) 2022 Jun 9 (pp. 1-6). IEEE.

[173] Tang Y, Zhao F, Yang Y, Lu H, Zhou Y, Xu B. Predicting vulnerable components via text mining or software metrics? An effort-aware perspective. In2015 IEEE International Conference on Software Quality, Reliability and Security 2015 Aug 3 (pp. 27-36). IEEE.

[174] Zhang Y, Lo D, Xia X, Xu B, Sun J, Li S. Combining software metrics and text features for vulnerable file prediction. In2015 20th International Conference on Engineering of Complex Computer Systems (ICECCS) 2015 Dec 9 (pp. 40-49). IEEE.

[175] Frey S, Rashid A, Anthonysamy P, Pinto-Albuquerque M, Naqvi SA. The good, the bad and the ugly: a study of security decisions in a cyber-physical systems game. IEEE Transactions on Software Engineering. 2017 Dec 13, 45(5):521-36.

[176] Shreeve B, Hallett J, Edwards M, Ramokapane KM, Atkins R, Rashid A. The best laid plans or lack thereof: Security decision-making of different stakeholder groups. IEEE Transactions on Software Engineering. 2020 Sep 14, 48(5):1515-28.

[177] Felderer M, Büchler M, Johns M, Brucker AD, Breu R, Pretschner A. Security testing: A survey. InAdvances in Computers 2016 Jan 1 (Vol. 101, pp. 1-51). Elsevier.

[178] Hussien ZA, Abdulmalik HA, Hussain MA, Nyangaresi VO, Ma J, Abduljabbar ZA, Abduljaleel IQ. Lightweight Integrity Preserving Scheme for Secure Data Exchange in Cloud-Based IoT Systems. Applied Sciences. 2023 Jan, 13(2):691.

[179] Mohammed NM, Niazi M, Alshayeb M, Mahmood S. Exploring software security approaches in software development lifecycle: A systematic mapping study. Computer Standards & Interfaces. 2017 Feb 1, 50:107-15.

[180] Do LN, Ali K, Livshits B, Bodden E, Smith J, Murphy-Hill E. Just-in-time static analysis. InProceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis 2017 Jul 10 (pp. 307-317).

[181] Nunes P, Medeiros I, Fonseca J, Neves N, Correia M, Vieira M. An empirical study on combining diverse static analysis tools for web security vulnerabilities based on development scenarios. Computing. 2019 Feb 12, 101:161-85.

[182] Myrbakken H, Colomo-Palacios RD. A multivocal literature review. Communications in Computer and Information Science, Mas, A., Mesquida, A., O'Connor, R., Rout, T., Dorling, A., Eds.:30-42.

[183] Buerkle A, Eaton W, Al-Yacoub A, Zimmer M, Kinnell P, Henshaw M, Coombes M, Chen WH, Lohse N. Towards industrial robots as a service (IRaaS): Flexibility, usability, safety and business models. Robotics and Computer-Integrated Manufacturing. 2023 Jun 1, 81:102484.

[184] Nyangaresi VO, Alsamhi SH. Towards secure traffic signaling in smart grids. In2021 3rd Global Power, Energy and Communication Conference (GPECOM) 2021 Oct 5 (pp. 196-201). IEEE.

[185] Jaatun MG. The building security in maturity model as a research tool. Empirical Research for Software Security: Foundations and Experience. 2017.

[186] Ansar SA, Khan RA. A phase-wise review of software security metrics. InNetworking Communication and Data Knowledge Engineering: Volume 2 2018 (pp. 15-25). Springer Singapore.

[187] Sentilles S, Papatheocharous E, Ciccozzi F. What Do We Know about Software Security Evaluation? A Preliminary Study. InQuASoQ@ APSEC 2018 (pp. 30-37).

[188] Morrison P, Moye D, Pandita R, Williams L. Mapping the field of software life cycle security metrics. Information and Software Technology. 2018 Oct 1, 102:146-59.

[189] Abood EW, Hussien ZA, Kawi HA, Abduljabbar ZA, Nyangaresi VO, Ma J, Al Sibahee MA, Kalafy A, Ahmad S. Provably secure and efficient audio compression based on compressive sensing. International Journal of Electrical & Computer Engineering (2088-8708). 2023 Feb 1, 13(1).

[190] Mihelič A, Vrhovec S, Hovelja T. Agile Development of Secure Software for Small and Medium-Sized Enterprises. Sustainability. 2023 Jan, 15(1):801.

[191] Rygge H, Jøsang A. Threat poker: solving security and privacy threats in agile software development. InSecure IT Systems: 23rd Nordic Conference, NordSec 2018, Oslo, Norway, November 28-30, 2018, Proceedings 23 2018 (pp. 468-483). Springer International Publishing.

[192] Nina H, Pow-Sang JA, Villavicencio M. Systematic mapping of the literature on secure software development. IEEE Access. 2021 Feb 26, 9:36852-67.

[193] Bishop D, Rowland P. Agile and secure software development: an unfinished story. Issues in Information Systems. 2019 Jan 1, 20(1).

[194] Tomanek M, Klima T. Penetration testing in agile software development projects. arXiv preprint arXiv:1504.00942. 2015 Apr 3.

[195] Nyangaresi VO, Abd-Elnaby M, Eid MM, NabihZakiRashed A. Trusted authority based session key agreement and authentication algorithm for smart grid networks. Transactions on Emerging Telecommunications Technologies. 2022 Sep, 33(9):e4528.

[196] Bukhsh FA, Bukhsh ZA, Daneva M. A systematic literature review on requirement prioritization techniques and their empirical evaluation. Computer Standards & Interfaces. 2020 Mar 1, 69:103389.

[197] Dabbagh M, Lee SP, Parizi RM. Functional and non-functional requirements prioritization: empirical evaluation of IPA, AHP-based, and HAM-based approaches. Soft computing. 2016 Nov, 20:4497-520.

[198] Santhosh Kumar SV, Selvi M, Kannan A. A Comprehensive Survey on Machine Learning-Based Intrusion Detection Systems for Secure Communication in Internet of Things. Computational Intelligence and Neuroscience. 2023 Jan 27, 2023.

[199] Aini Q, Harahap EP, Santoso NP, Sari SN, Sunarya PA. Blockchain Based Certificate Verification System Management. APTISI Transactions on Management (ATM). 2023, 7(3):1-0.

[200] Rajesh M, Acharya TA, Hajiyev H, Lydia EL, Alshahrani HM, Nour MK, Mohamed A, Al Duhayyim M. Blockchain Driven Metaheuristic Route Planning in Secure Wireless Sensor Networks. CMC-computers materials & continua. 2023 Jan 1, 74(1):933-49.