Global Journal of Engineering and Technology Advances

eISSN: 2582-5003

Cross Ref DOI: 10.30574/gjeta

Journal homepage: https://gjeta.com/

(Review Article)

Check for updates

An investigation into software quality and security: Past research works and existing gaps

Fridah Chepkemoi Korir *

Jomo Kenyatta University of Agriculture and Technology, Juja, Kenya.

Global Journal of Engineering and Technology Advances, 2023, 14(03), 149–171

Publication history: Received on 16 February 2023; revised on 28 March 2023; accepted on 31 March 2023

Article DOI: https://doi.org/10.30574/gjeta.2023.14.3.0059

Abstract

Software security is concerned with the protection of data, facilities and applications from harm that may be occasioned by malware attacks such as password sniffing, viruses and hijacking. It is a system-wide concept that takes into account both security mechanisms such as access control as well as the design for security, such as a robust design that renders software attack complicated. It may encompass building of secure software, which comprises of the designing of software to be attack-resistant, ensuring that software is error-free, and educating software developers, architects, and users about the building of secure artifacts. In this regard, insecure software negatively affects organization's reputations with customers, partners, and investors. The goal of this paper is to investigate some of the issues that make the software insecure, as well as the approaches that have been developed to boost software quality and security. The outcomes indicate that various models, techniques, frameworks and approaches to software quality have been developed over the recent past. However, only a few of them give reliable evidence for creating secure software applications.

Keywords: Attacks; Frameworks; Models; Quality; Security; Software

1. Introduction

Software systems have been extensively deployment in various domains and have become an integral part of human life. Most of these software systems process large and critical data which needs to be secured [1]. In addition, they are required to satisfy user needs or functional requirements. The rapid developments in information and communication technologies (ICTs) have made software security a key concern. Such developments include Internet of Things (IoT), Internet of Every Things (IoE), the advancement of Internet-based software systems, cloud computing, social networking, and location-based services. Moreover, new business paradigms, versatile customer requirements, rapid advancement in ICTs, and new regulations are making a software application evolve [2], [3]. In this complex software deployment scenarios, misuse of software [4], [5] can lead to various outcomes, such as sabotage in the communication sector, heavy economic loss in the financial sector, critical data theft in databases [6], as well as misuse of software in the missile controlling systems. All these outcomes have the potential of endangering human life. As explained in [7], software security features play a critical role in the security designs. These features help to enhance software security as well as helping to uphold software quality [8]. However, according to [9] and [10], software security ensures that the Confidentiality, Integrity, and Availability (CIA) of data and services are not interfered with. This can be achieved when security is considered during all Software Development Life Cycle (SDLC) phases.

Software security deals with the protection of data, facilities and applications from harm caused by malware attacks such as password sniffing, viruses and hijacking. The malicious activities are mounted by various types of attackers, including hackers, crackers, domestic cyber-terrorists, industrial spies and international military [11], [12], [13]. As

*Corresponding author: Fridah Chepkemoi Korir

pointed out in [14], software security is the capability of the software to resist, tolerate and recover from events that threaten its dependability. However, authors in [15] treat software security as the process of evaluating an application to discover risks and vulnerabilities of this application and its data [16], [17]. It is a system-wide concept that takes into account both security mechanisms such as access control as well as the design for security, such as a robust design [18] that makes software attack difficult. Basically, it involves the building of secure software, which comprises of the designing of software to be secure, ensuring that software is secure [19], and educating software developers, architects, and users about the building of secure artifacts. In so doing, it defends against software exploit by getting by the design right and avoiding common mistakes [20]. As explained in [21], secure software deals with the building of software that can withstand strong attacks, as well as maintaining basic security structures such as confidentiality, integrity, and access to sensitive assets. These three security structures are referred to as the CIA such that any software that enlists the CIA can be considered as secure software. With regard to software security characteristics, authors in [22] have defined it as the degree to which a product or system protects information and data. This means that people, other products or systems have the degree of data access that is appropriate to their types and levels of authorization [23]. It should also preserve features of secure software such as confidentiality, integrity, accountability and authenticity [24].

The authors in [25] have identified security as one of the most critical aspects of software quality. This is because software security incorporates processes that create and develop software that assures the integrity, confidentiality, and availability of its code, data, and services [26], [27], [28], [29]. As software development becomes more complex, distributed, and concurrent, security issues have been noted to have greater influence on the quality of the developed software [30]. From the empirical software engineering perspective, metrics of developer behavior such as unfocussed contribution [31], different development priorities [32], code complexity [33], [34] and large code changes [35] are often deployed to explain code quality. Unfortunately, the deployed software is continuously under the attack of hackers who exploit vulnerabilities. Over the recent past, there has been an increase in these attacks [36]-[39]. As discussed in [40] and [41], the elimination of bugs in the form of buffer overload and incompatible error management are major issues in software security [42]. To incorporate security into the software engineering paradigm, it should be put into consideration from the start of the SDLC [43]-[47]. This has led to the emergence of Secure Software Engineering (SSE) concept, which deals with the process of designing, building, and testing software. This ensures that this software is secure. It includes secure SDLC processes and secure software development (SSD) methods [48], [49],[50], [51].

As explained in [52], a software project might have different development practices, depending on its size. For instance, big software projects may involve developers working in parallel to increase the speed of development [53-[56]. In addressing both the technological and human aspects that may be involved, there is need to understand SSE methods. Basically, SSE is concerned with the building of software that can withstand potentially aggressive attacks. In addition, SSE encompasses maintaining basic security features such as privacy, integrity, and access to sensitive assets [57]. As the best practice, SSE recognizes that software security is a crucial factor that needs to be taken into account during the start of the software life cycle [58]-[61]. Since security problems in the SDLC are difficult to address, SSE has become a significant paradigm in the development of secure software for the software industry in recent years. This is supported by the authors in [62] and [63] who explain that during the entire software development life cycle, software security is an essential factor that needs to be addressed. Unfortunately, security is characterized as a non-functional requirement, and hence security checks [64] are usually carried out during the final phases of SDLC. It is therefore important for software security to be taken care of even in the first phase of the software development process. In a nutshell, software security is the key to the software's success, especially in today's fast-paced and technology-oriented world. Therefore, the incorporation of security at any level of the SDLC has become an urgent requirement. Unfortunately, software development organizations regard security as an afterthought issue, and hence continue to face security threats [25], [65], [66], [67], [68], [69], [70]. The goals of this current work include the following:

- Offer a review of the various causes of software vulnerabilities
- Establish the rationale behind the rising need for software quality and security
- Investigate the various approaches geared towards software quality and security

The rest of this paper is structured as follows: Section 2 discusses the various causes of software vulnerabilities while Section 3 expounds on the rising need for software quality and security. On the other hand, Section 4 explains the various approaches geared towards software quality and security while Section 5 presents some research gaps and future research directions. Finally, Section 6 concludes this paper.

2. Causes of software vulnerabilities

One of the main reasons for widespread vulnerabilities [71] is failure to make security a key priority [10]. This is because even diligent businesses use the fix and penetrate approach in which security is accessed after completing the project [72]-[74]. The drawback of this method is that the application users do not apply these patches. As pointed out in [75] and [76], a secure software should not be accessed, updated, or targeted by any unauthorized users [77]. As such, software that has no vulnerabilities [78] is considered highly stable, whereas software that has at least one vulnerability is considered vulnerable. As pointed out in [79], SDLC endeavors to produce high-quality and low-cost applications in the shortest amount of time. This is achieved by offering a well structured step flow that aids enterprises in easily produce high-quality, well-tested, and ready-to-use production of software [80]-[82]. The common phases of SDLC include requirement, design, coding, testing, deployment, and maintenance [83]. Basically, all these phases depend on each other are of equal importance. Therefore, if security is not incorporated during these phases of SDLC, then the resultant product will be vulnerable to security threats [84], [85]. To counter this, secure SDLC processes must be followed to ensure that security-related activities are an integral part of the overall development effort [75], [86], [87]. However, the authors in [88] have explained that security protection is not considered in the overall system development lifecycle and hence numerous security breaches can occur [89], [90], [91], [92].

The authors in [62] and [93]-[96] have identified a number of software security issues during the coding phase of SDLC. On the other hand, some of the most common malware attacks include viruses, trojan virus, brute force attack, DNS hijacking, replay attacks [97], denial of service, flooding attacks, slicing attacks and cookie poisoning. These attacks negatively affect the processes of secure software development [11], [13], [98], [99]. In MITRE's Common Vulnerabilities Exposures database, the latest classification of common defects by type is provided. Here, the most common forms of security vulnerabilities are identified as weak encryption, explicit password storage, insecure communication [100], and synchronization errors [101]. However, the authors in [13] and [102] have identified invalidated redirects and forwards, improper use of secure APIs [103], weak encryption, insecure communication, man in the middle [104], and bandwidth usage are some of the most common security issues that hamper the communication and encryption processes.

During the software development process, majority of the security attacks are possible due to implementation flaws such as improper input validation [105], improper authentication and authorization mechanisms [106], improper session management [107], and other vulnerabilities such as Session-Id vulnerable or theft, incorrectly implemented logouts, lock failed attempts per browser session, peer-user session restriction, and log replay feature. All these mishaps compromise the application's intended functionality [11], [48], [108]. However, spoofing [109], tampering, repudiation [110], information disclosure, denial of services [111], elevation of privilege and failure to restrict uniform resource locator (URL) access are some of the most common security issues that hamper the process of secure authorization and authentication [76], [112], [113], [114], [115], [116], [117]. Cross-site scripting, cross-site request forgery, format string problems, code and command injection, auto-complete attribute not enabled have been noted to be some of the software security risks in the deployment phase. On the other hand, software security risks in maintenance phase have been identified as POST change requests for GET, POST directives with invalidated parameters, as well as a database injection vulnerabilities [48], [93], [108], [118]-[123]. Here, incorrect input validation [124] refers to the lack of or incorrect substantiation of input provided by a user via the application's user interface. On the other hand, injection attacks [125] take advantage of the lack of input validation controls to allow malicious inputs to be passed in, which can be used to obtain elevated rights, alter data, or crash a system [126]-[128]. On the other hand, code injection attacks can breach data security, cause a loss of services and harm thousands of users' systems [118].

The vulnerabilities in software systems have been noted to include outdated software or fimware, default usernames and password, password conjuncture, and the inability to run software updates or change usernames and passwords. These credentials are leveraged to gain initial access to systems of corporate targets which can then be further exploited [50], [102], [129]. As explained in [130], software testing is the most time-consuming, complicated, and costly process of the SDLC. This phase is an important component of improving the efficiency [131] of software development projects [1]. Although it is an essential part of software development, rigorous testing is not normally the focus of software engineering education [132]. As a result, software developers often regard software testing as a liability, lowering overall software quality [133]-[135]. According to [136], developing secure software systems is dogged by many challenges. These include designing authentication protocols [137], improper configuration management, building strong cryptosystems [138], devising effective trust models and security policies [139]. Configuration management is an important component during secure maintenance and operation phase [140]. Some of the common software security risks which affect deployment phase of the SDLC have been identified in [1], [48], [62], [136], [140], [142].

The authors in [143] have defined threat modeling as a systematic method for identifying threats that may compromise security [144]. As such, it is considered a well-known accepted practice by the software testing industry [145]. The aim of this phase is to find possible bugs and errors in the system and eliminate them. Some of the software security risks during software testing phase of SDLC have been discussed in [48], [86], [146], [147], [148], [149]. As explained in [150], software development iterations are of limited time, often lasting for few weeks. This makes fitting security activities such as security requirement elicitation challenging since they are often time-consuming [151]. In addition, defining security policies takes time, which raises the cost of software development. Some of the common issues due to time pressure in the secure software development process are identified in [118], [141], [150].

The authors in [152] point out that vulnerabilities of design level works as the major sources of security risks in software systems. In fact, 50% of software defects are normally identified and detected in the designing phase of SDLC [153]-[155]. As such, reducing the risks at this phase may minimize the efforts in other phases.

3. The rising need for software quality and security

Insecure software negatively affects organization’s reputations with customers, partners, and investors. It can also increases costs as organizations are forced to repair unreliable applications. In so doing, it can potentially delay other development efforts as limited resources [156] are assigned to address current software deficiencies [29]. The current literature on requirement security has yielded different security risks that might occur if security is not incorporated from the beginning [157]. For instance, some security risks inherent in the requirement phase of SDLC are discussed in [10], [83], [112], [150], [158], [159]. In addition, design flaws have been noted to be one of the most common sources of security threats [160] in software systems [75], [161]. As pointed out in [162], most of the software bugs are discovered during the design process of the SDLC. This is because the design process of the SDLC serves as the foundation for designing a secure software system [163]. Some of the most common security challenges encountered during software design have been identified in [50], [75], [161], [113], [115], [164]. As such, reducing risks in this step can minimize the effort needed in subsequent phases [1], [165].

Each phase of the SDLC must incorporate appropriate security protections [166], analyses, and countermeasures that result in more secure code being released [94], [167]. As discussed in [168] and [169], the current trend is for developers to import functionality from third party free open-source software (FoSS) libraries by including them into their projects as dependencies [168], [169]. Such software engineering practice permits developers to use FoSS libraries as building blocks. This can potentially reduce development cost and time. According to [170], even for proprietary software, the fraction of homegrown code decreased to 5%. The reports from the software industry show that third party code inherited through dependencies is four times larger than the size of the own code base as an industry average [171]. In today’s software ecosystem, homegrown code is only a fraction of the total code base that is shipped to customers [171], [172]. However, a large leverage means that several libraries are deployed, which may require integration and update costs. In addition, developers rarely update the third party libraries they are using [173], [174]. This is attributed to the possibility of introducing incompatible, breaking changes [175]. Using many libraries increases the attack surface, and third-party libraries are known to introduce functionality bugs and security vulnerabilities [176] into the projects that use them [174], [177]. In some cases, dependent projects keep using outdated components for a decade or more [178] thus increasing the window of possible exploitations.

Literature has shown that developers have a habit of reacting to the issues connected with their own code of their libraries or their direct dependencies [175], [179]. However, transitive dependencies are known to introduce security vulnerabilities to some extent [171], [177], [180]. A number of technical studies [173], [174], [177], [180], [181] have shown that FOSS dependencies, although being widely used by both commercial and FOSS projects, are not often maintained properly. For instance, large share of projects have outdated dependencies.

4. Approaches to software quality and security

Various models, techniques, frameworks and approaches to software quality have been developed over the recent past. These include Capability Maturity Model Integration (CMMI), Microsoft Software Development Life Cycle (MS-SDL), misuse case modeling, abuse case modeling, Knowledge Acquisition for Automated Specification, System Security Engineering Capability Maturity Model (SSE-CMM), Open Web Application Security Project (OWASP), and Secure Tropos Methodology [165]. Apart from these approaches, security testing technique has been identified as one of the most significant, effective, and commonly applied measures for the improvement of software security. It has been employed to identify the vulnerabilities and to ensure the functionality of security. For the identification of threats [182] that might compromise security, threat modeling is deployed. As explained in [108], improper authentication and authorization mechanisms refer to the erroneous implementation of authentication functions and access-control policies. In this regard, authentication and authorization are critical components of basic security processes. As discussed in [183], these two concepts are particularly important in the production of secure software.

The focus of conventional security mechanisms is on network systems. Many organizations spend huge amounts of money to make their network secure. These mechanisms include Intrusion detection system (IDS), firewalls, encryption, antivirus, and antispyware [48], [83], [184]. As explained in [86], building secure software means developing software that functions properly even under malicious attacks. This includes addressing the security challenges through the whole SDLC, especially in the early stages during the design phase [185]. The outcome is the reduction in the risk of overlooking critical security requirements or introducing security flaws throughout the implementation process. To build and deploy a secure software system, there is need for the integration of security features into the life cycle of application development and align current SSE methods [186], [187]. However, most organizations view security as a post-development process, and hence security is not considered during the pre-development phase. Consequently, there is no approval for the method to be used, and hence there is little understanding of the need for secure software development [188]. There are also few facts about the effectiveness [189] of existing approaches to dealing with real problems. In addition, there is limited view of how the existing approaches contribute to the assessment of safety concerns [190].

According to [191], threats put systems at greater risk for major losses that can be difficult to recover. The majority of software programs are designed and deployed without attention to protection desires [192], [193]. Hidden attack risks within or outside the organization are emerging day-by-day, resulting in huge financial loss, as well as confidentiality [194] and credibility losses. This is because they put the availability and integrity of organizational data at risk. The authors in [191] and [195] explain that the coding phase of SDLC is more prone to errors, as the programmer leaves some bugs unintentionally. This increases software vulnerability to more attacks. Such vulnerabilities may include denial of services, code execution, memory corruption, data loss, cross-site scripting, improper access control, SQL injection, integer overflow and buffer overflow [196]. To curb these issues, researchers in the software industry have adopted a wide variety of software security practices, approaches, and methods [197], [198], [199], [3], [200].

Several companies have created maturity models and frameworks to assess the degree of maturity of their software security practices. For instance, Correctness by Construction is a technique for developing high integrity software [201]. The seven main principles of Correctness by Construction include the following: expect requirements to change, know why you are testing, eliminate errors before testing, write software that is easy to verify [202], develop incrementally, some aspects of software development are just plain hard, the software is not useful by itself. On the other hand, the authors in [203] and [204] recommend seven touchpoint operations, which include abuse cases; security requirements; architectural risk analysis; code review and repair; penetration testing; and security operations. The aim of these touchpoints is to create secure software, all of which are connected to software development artifacts. Similarly, Microsoft has developed the Microsoft Trustworthy Computing Security Development Lifecycle [205], which adds a set of security practices to each step of its software development process. On the other hand, Secure Software Development Process Model (S2D-ProM) [190] has been developed to act as a strategy-oriented process model that offers guidance and support to developers and software engineers at all level, from beginners to experts, to build secure software. Similarly, TSP Secure (Team Software Process for Secure Software Development) [206] has been developed specifically for software teams. It endeavors to help them create a high-performance team and prepare their work to produce the best results. The TSP Secure focuses directly on the security of software [207] in three ways: planning, development and management, and training for developers about security-related aspects and other team members.

As explained in [208], Comprehensive, Lightweight Application Security Process (CLASP) is a straightforward process that consists of 24 high-level security activities that can be completely or partially integrated into software during the SDLC. In CLASP threat modeling and risk analysis [209] is performed during requirement and design phase. In the design and implementation phase, it suggests secure design guidelines and secure coding standards [210], [211], [212], [213]. Inspections, static code analysis, and security testing [214] are performed in the assurance phase [215]. On the other hand, authors in [75] have conducted a Multi-vocal literature review to identify the best practices for designing secure software. Based on identified best practices, a framework Secure Software Design Maturity Model (SSDMM) was developed. Similarly, the Security Quality Requirements Engineering (SQUARE) methodology has been developed to facilitate elicitation, classification, and prioritization of security specifications for information technology systems and applications [216]. In addition, Appropriate and Effective Guidance for Information Security (AEGIS) has been developed to evaluate device assets and their relationships. Thereafter, it moves on to risk analysis, which defines weaknesses, threats, and risks [217], [218]. According to [219], the Secure Software Development Model (SSDM) security training offers stakeholders in software development with adequate security education [220]. During the requirements process of SSDM, a threat model is used to identify and their capabilities.

As discussed in [113], Microsoft uses STRIDE to model threats to their systems. Here, threats are defined by looking into the possibilities of spoofing identity, tampering with data, repudiation, information leakage, denial of services [221], and elevation in the given situation. The authors in [222] explain that numerous security approaches have been developed to assist the software engineers in evaluating security risks, such approaches include Attack Trees, combining goal-orientation and use-case modeling (an effective method of software requirement engineering) [223] and Secure Tropos (a security-oriented extension to the goal-driven requirements engineering methodology) [224]. Other approaches allow the software engineers to address these risks by reusing design decisions [225] or sustaining the decision making process [226]. Other software security approaches are McGraw's Secure Software Development Life Cycle (SSDLC) process [40], Microsoft Software Development Life Cycle (SDL) or Trustworthy Computing Security Development Life Cycle, Security Requirements Engineering Process (SREP) [227], Aprville and Pourzandi's Secure Software Development Life Cycle process, Core security requirements artifacts [228], Comprehensive, Lightweight Application Security Process (CLASP), Haley framework [229], and Security Quality Requirements Engineering (SQUARE). In addition, the authors in [208] explain that OWASP Security Verification Standard (ASVS) version 3.0 is a community effort to establish a framework of security requirements and controls [230] that focus on normalizing the functional and non-functional security controls required when designing, developing, and testing modern web applications. Basically, the ASVS comprises of a list of application security requirements or tests used by architects, developers, testers, security professionals, and even consumers to define what a secure application is.

On the other hand, ISO/IEC 27001:2005 covers all types of organizations such as commercial enterprises, government agencies, and non-profit organizations [231], [232], [233]. It specifies the requirements for establishing, implementing, operating, monitoring, reviewing, maintaining, and improving a documented Information Security Management System [234] within the context of the organization's overall business risks. In addition, it stipulates requirements for the implementation of security controls [235], [236] customized to the needs of individual organizations or parts thereof. Its design permits the selection of adequate and proportionate security controls that protect information assets and give confidence to interested parties. On their part, the authors in [237] explain that browser identity indicators such as uniform resource locators (URLs) and certificates help users identify phishing, social engineering, and other attacks. However, previous lab studies and surveys have suggested that older browser identity UIs are not effective security tools. Modern browser identity indicators have also been noted to be ineffective. Therefore, to design better identity indicators, browsers need to focus on active negative indicators, explore using prominent UI as an opportunity for user education, and incorporate user research into the design phase. Such goals have been achieved by the works in [197],[198] and [199]. However, most of these studies address only maintenance, evolution, implementation and feedback phases. The authors in [87] point out that the requirement stage in the SDLC is the primary stage where the initial plan for software is made. It necessitates a set of initial specifications, which are collected from various sources. To accomplish this, a number of methods such as brainstorming, group sessions, and interviews are used.

The aim of Secure requirement engineering (SRE) is to offer complete security by implementing basic security functions such as confidentiality, integrity, and availability. This phase involves activities such as security requirements identification and inception, documentation, elicitation, analysis and negotiation, mapping, verification and validation, prioritization and management, authentication, and authorization [10], [112], [238]. The commonly deployed best practices for handling security issues during the requirement stage of SDLC have been highlighted in [10], [86], [83], [Keshta et al., 2017], [112], [159], [98], [238], [239], [240]. As explained in [1] and [162], the design phase is one of the most creative stages of the SDLC, and is therefore important from the viewpoint of security [241]. The authors in [1] have identified design-level flaws as the most common sources of security risks in software systems, where 50% of the software defects are identified and detected during this phase. Here, the security design architecture stipulates design methods such a strongly typed programming, least privilege, develop threat modeling, analyze and minimize attack surface. As such, the software developer must consider security best practices during design in a manner that is appropriate and secure. Some of the most widely used design security practices that should be followed when designing secure software have been discussed in [1], [86], [76], [29], [161], [162], [113], [115], [164], [239], [242].

The authors in [29] have pointed out that 80% of system penetration is due to coding errors in commercial software. Increased bugs, security issues, and costs are all associated with bad code. Due to time-to-market pressures, software developers are under pressure to meet deadlines. In addition, there is lack of security expertise and developers fail to follow secure code guidelines. An assumption made here is that perimeter security is sufficient to protect applications. To address this issue, security code reviews need to be conducted while the code is being checked for functionality, whether manual or automated. The goal here is to verify the fundamental tenets of software security [86], [243]. In addition, the programmers must be aware of implementation-level vulnerabilities when writing secure code and they must utilize the documentation and guidelines created in earlier stages to help them write secure code. As such, the authors in [48], [29], [86], [149], [239], [244], [245] and [246] have discussed some of the prescriptive actions to increase security during the coding phase of SDLC. As explained in [130], software testing is the most time-consuming, complex, and costly phase of the SDLC, whose goal is to identify and fix any bugs or errors in the system. Here, security testers employ misuse cases, threat models and design documents to detect potential attacks and the consequences of successful attacks. Upon the completion of security testing, test documents containing security test cases [247] and a prioritized list of vulnerabilities resulting from automated and manual dynamic analysis are created. In this regard, some of the prescriptive actions to increase security during the testing phase of SDLC are described in [48], [86], [87], [146], [147], [148], [149], [239], [248], [249].

As discussed in [99] and [158], after the software is deployed into its operational environment, it is critical to monitor responses to flaws and vulnerabilities of the system to check for new evolved security patterns. After the identification of new security patterns, the same should be included in the requirement stage for further security improvements in subsequent releases. Here, static analysis and peer review are two useful procedures for mitigating or minimizing newly discovered vulnerabilities [250], [251]. Thereafter, final security reviews and audits are performed during the secure deployment phase, in which customer satisfaction is vital. Some of the prescriptive actions to increase security [252] during the deployment phase of SDLC are identified in [48], [29], [149], [239], [253], [254]. Before deploying software, administrators must understand the software's security stance such that some of the identified faults that were not addressed previously are revisited, prioritized, and corrected after deployment. This is followed by the tracking of new threats by the maintenance team such that they are addressed promptly to prevent security breaches [118]. Some of the approaches to increase security during the maintenance phase of SDLC are identified in [29], [150], [239], [253], [255], [256]. As discussed in [15], security activities during the requirement phase serve three purposes. To start with, initial security requirements [257], [258] are identified and implemented. Secondly, with the security requirements in hand, the project team understands and recognizes the importance of security. Thirdly, with the needs of security in the hands, budget, resources, and time of security activities in future stages can be better estimated.

The authors in [259], [260] explain that during the design phase, the project team focuses on identifying the attacker's interests, potential access points, and critical security areas. This is followed by the identification of threats running on the software. Basically, all the security data collected in the design phase goes into the threat models, which are important milestone in terms of secure software. This involves gauging whether the security building function offers full details of how the software can be attacked, the asset that is likely to be attacked, the areas of attack that are attractive, and the kind of threats [7]. Based on this information, the security structure is continuously updated to cater for new threats. As explained in [261], the implementation phase plays a twofold role from a security perspective. To start with, it prevents security errors entering the software. Secondly, it detects existing software errors. Here, the first role is accomplished by writing a secure code while the second role of detecting security errors begins with static analysis by automated tools [262]. After automatic analysis, a manual update is performed. Thereafter, the software is fully functional and ready to go to the testing phase. According to [263], tests are performed mainly on test cases generated during test planning. Here, the testing team identifies security errors [264], [265], reports to the development team, and the development team corrects them in this code. The testing phase ends when all test cases are conducted, and retrospective testing of all sensitive areas has taken place [266]. Similar to other forms of testing, security testing involves the determination of who should do it and what activities should be undertaken.

Before the release of the software, a security reviews must be performed [267] to identify the remaining security errors. Thereafter, the development team corrects code against security errors identified in the review report. After a review, a security audit is conducted, and based on such audit reports, management decided to release the software [268]. Upon release and distribution, the software is commercially used, but decisions may be made later to rectify non-critical safety errors. This may involve changing the code to remove these security errors in the form of a patch [269], [270]. After rigorous testing, the patch is applied to the software, which is the followed by the release of the patch [271]. To address the software system security, various models [272], practices, strategies, and methods have been proposed. They have been shown to improve security procedures in the stages of SDLC [273]. To effectively address security issues that exist during the application process, it is necessary to consider secure procedures in all development processes. This helps to minimize the threats of critical security requirements or to identify critical errors in software development [274], [275]. It has been shown that security is often neglected during software development. However, there is a growing emphasis to include the security aspects in every phase of software development, specifically at the early phases.

The authors in [276] have developed a Security-Requirements-Elicitation and- Assessment-Mechanism (SecREAM) to facilitate holding of the security issues [277] that appear at the start of software development. On the other hand, the authors in [152] have provided a mechanism for measuring the security requirements engineering process. This mechanism is aligned with the method of SQUARE. Similarly, the authors in [278] have proposed a stochastic type maintenance method for the security of software through the use of a closed queuing-model of unreliable backups. However, a Software Security Assurance Model (SSAM) has been developed in [1] to assist vendor organizations to assess their readiness for secure software development. Similarly, the authors in [279] proposed a standards-setting approach to software product and software supply network modeling. Despite the fact that this allows developers to anticipate upcoming changes in the software ecosystems, the approach aims at development within one company. As such, it does not suit the purpose of modeling FoSS infrastructure. On the other hand, the authors in [280] have proposed a simple model to help software developers to decide whether to include FoSS components into their projects. This model estimates the value of FOSS libraries based on the possibility of receiving additional support from the developers of an FOSS community.

5. Research gaps and future research directions

It is clear that several methodologies, strategies, and models have been proposed and developed to address software security. However, it is evident that only a few of them give reliable evidence for creating secure software applications. Therefore, software security issues have not been adequately addressed, and hence integrating security procedures into the SDLC remains a challenge. It has also been noted that hidden attack risks within or outside the organization are emerging day by day. This has resulted in huge financial loss, as well as confidentiality and credibility losses. This is because it puts the availability and integrity of organizational data at risk [281], [282], [283]. As pointed out in [284], most businesses view security as a post-development process. Consequently, security is not considered at some point in the predevelopment phase [285]. It has also been noted that many software development companies do not follow best practices to incorporate security in SDLC [286], [93]. This negligence includes lack of awareness, fear of time and cost overruns. The other reasons include the fact that the development teams are always in a hurry, use of third-party components and lack of qualified professionals. In some cases, majority of software programs are designed and deployed without attention to protection mechanisms [287], [288], [289]. Over the recent past, various approaches to software quality have been developed. These include CMMI, MS-SDL, misuse case modeling, abuse case modeling, knowledge acquisition for automated specification, SSE-CMM, OWASP and Secure Tropos Methodology [165]. However, there exists no explicit solution for incorporating security into all phases of SDLC.

The authors in [3], [197], [198], [199], [200] have introduced a wide variety of software security practices, approaches, and methods. In addition, several companies have created maturity models and frameworks to assess the degree of maturity of their software security practices. However, none of these models or structures is specifically committed to recognizing security risks, threats and their practices in the SDLC. Consequently, they fall short of covering all aspects and activities of a secure SDLC. To curb this, it is critical to recognize the security threats that vendor organizations face while developing secure applications so as to develop risk mitigation strategies. This will enable software development vendors to assess their maturity and assurance levels, as well as improve their secure SDLC performance. In addition, it will help raise the level of awareness among software engineers. As pointed out in [99], vulnerability oriented architectural research offers a systematic and methodical approach to evaluating a wide variety of possible vulnerabilities. However, it is time-consuming and costly. To estimate the severity and cost of security threats, some maintenance and stakeholder considerations have been identified in [62], [99], [290]. As explained in [291], SSE postulates that software security is a critical factor that should be assessed early in the SDLC process. To build and deploy a secure software system, there is need to integrate security features into application development life cycle and adapt the latest SSE practices [43], [44].

Many software security issues stem from insufficient or incorrect identification, documentation, analysis, mapping, prioritization, specification and availability of security requirements. In this environment, the importance of identifying non-functional security requirements should be stressed. This is because it helps in the reduction or elimination of software vulnerabilities [10], [292], [140]. Misuse cases are other issues that need to be addressed. These cases, similar to use cases, they specify what a system should not do. They represent a great way to get security requirements [140], [141], [142], [148]. It has been noted that conventional security solutions such as antivirus, intrusion detection mechanisms, and firewalls are not enough to reduce the risk in the coding phase of the SDLC. Therefore, there is need for various suitable security defenses, practices, analysis, and countermeasures that can boost the security of the released code [94], [167]. Although patches have been developed to address the software flaws, software remains vulnerable to a variety of security threats. As such, there is need to monitor responses to flaws and vulnerabilities of the system so as to discover newly evolved security threats. After identifying these new security risks, they should be included in the requirement stage for further security improvements in subsequent releases [99], [158]. In this regard, static analysis and peer review are two useful procedures for mitigating or minimizing newly discovered vulnerabilities [29]. However, final security reviews and audits should be performed during the secure deployment phase [293], [294].

Based on the reviewed literature, many software development techniques do not explicitly include software security measures during software development as they move from demand engineering to their final products. As such, the integration of software security at each stage of the software development life cycle has become an urgent need. To tackle software security, numerous methods, techniques, and models have been suggested and developed.

Unfortunately, only a few of them offer strong evidence for building secure software applications. Due to budget constraints and shorter software release time in the market, many developers consider security as a subsequent thinking problem that may contribute to poor software quality. Although software security was considered part of software testing in the early days, it has over time been shown that security is not a serious concern. It is therefore important to consider how software engineers can incorporate security into the early stage of SDLC. It has been shown that the deployed software is under continuous attack from hackers exploiting vulnerabilities for decades. Consequently, there has been an increase in these attacks. There is therefore need to incorporate various suitable security defenses, analysis [295], and countermeasures in each phase of SDLC that can further secure the released code.

Further, there is need to incorporate strong and latest security features into application development life cycle and adapt the latest SSE approaches to build and deploy a secure software system [296]. As illustrated by a quantitative study on Android libraries, updating software is not always a technically feasible solution [297]. This is because almost every library update breaks the dependent project, as explained in [175]. The presence of such dilemma may require the identification of alternative solutions to software updates. It has been shown that software testing is the most lengthy, complex, and expensive phase of SDLC [298]. It is a vital activity that is geared towards increasing the quality of software development projects. Although it is a core phase for software development, the thorough testing of the programs is not always the core subject under software engineering education. Consequently, the software developers often treat software testing as a liability. This negatively affects the overall quality of software. The main reason is that standardized testing mechanisms are recurrently regarded as boring and challenging when compared to the creative coding phase and design activities.

6. Conclusion

The rapid developments in information and communication technologies have made software security a key concern. Such developments include IoT, IoE, the advancement of Internet-based software systems, cloud computing, social networking, and location-based services. In this complex software deployment scenarios, misuse of software can lead to various outcomes, such as sabotage in the communication sector, heavy economic loss in the financial sector, critical data theft in database, as well as misuse of software in the missile controlling systems. Failure to make security a key priority has been noted to be one of the main reasons for widespread vulnerabilities. As such, models, techniques, frameworks and approaches to software quality have been developed, exampled by CMMI, MS-SDL, misuse case modeling, abuse case modeling, knowledge acquisition for automated specification, SSE-CMM, OWASP, and secure tropos methodology. Unfortunately, only a few of these approaches give reliable evidence for creating secure software applications. As such, software security issues have not been adequately addressed, and hence integrating security procedures into the SDLC remains a challenge. Future work will involve the development of practical solutions to address both software quality and security.

Compliance with ethical standards

Acknowledgments

My appreciation goes to my supervisors and colleagues who provided some help when writing this paper.

References

[1] Khan RA, Khan SU, Khan HU, Ilyas M. Systematic mapping study on security approaches in secure software engineering. IEEE Access. 2021 Jan 18, 9:19139-60.

[2] Szczepaniuk EK, Szczepaniuk H, Rokicki T, Klepacki B. Information security assessment in public administration. Computers & Security. 2020 Mar 1, 90:101709.

[3] Meshram C, Alsanad A, Tembhurne JV, Shende SW, Kalare KW, Meshram SG, Akbar MA, Gumaei A. A provably secure lightweight subtree-based short signature scheme with fuzzy user data sharing for human-centered IoT. IEEE Access. 2020 Dec 21, 9:3649-59.

[4] Kalloniatis C, Mouratidis H, Islam S. Evaluating cloud deployment scenarios based on security and privacy requirements. Requirements Engineering. 2013 Nov, 18:299-319.

[5] Qin Z, Denker G, Giannelli C, Bellavista P, Venkatasubramanian N. A software defined networking architecture for the internet-of-things. In2014 IEEE network operations and management symposium (NOMS) 2014 May 5 (pp. 1-9). IEEE.

[6] 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.

[7] Kumar R, Khan SA, Khan RA. Analytical network process for software security: a design perspective. CSI transactions on ICT. 2016 Dec, 4:255-8.

[8] Moyo S, Mnkandla E. A novel lightweight solo software development methodology with optimum security practices. IEEE Access. 2020 Feb 3, 8:33735-47.

[9] Tatam M, Shanmugam B, Azam S, Kannoorpatti K. A review of threat modelling approaches for APT-style attacks. Heliyon. 2021 Jan 1, 7(1):e05969.

[10] Niazi M, Saeed AM, Alshayeb M, Mahmood S, Zafar S. A maturity model for secure requirements engineering. Computers & Security. 2020 Aug 1, 95:101852.

[11] Baca D, Petersen K. Countermeasure graphs for software security risk assessment: An action research. Journal of Systems and Software. 2013 Sep 1, 86(9):2411-28.

[12] 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.

[13] Masood A, Java J. Static analysis for web service security-Tools & techniques for a secure development life cycle. In2015 IEEE International Symposium on Technologies for Homeland Security (HST) 2015 Apr 14 (pp. 1-6). IEEE.

[14] Hatzivasilis G, Papaefstathiou I, Manifavas C. Software security, privacy, and dependability: Metrics and measurement. IEEE Software. 2016 Mar 18, 33(4):46-54.

[15] Parizi RM, Qian K, Shahriar H, Wu F, Tao L. Benchmark requirements for assessing software security vulnerability testing tools. In2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC) 2018 Jul 23 (Vol. 1, pp. 825-826). IEEE.

[16] Zhang S, Ou X, Caragea D. Predicting cyber risks through national vulnerability database. Information Security Journal: A Global Perspective. 2015 Dec 31, 24(4-6):194-206.

[17] Cirnu CE, Rotună CI, Vevera AV, Boncea R. Measures to mitigate cybersecurity risks and vulnerabilities in service-oriented architecture. Studies in Informatics and Control. 2018 Sep 1, 27(3):359-68.

[18] Omollo VN, Musyoki S. Global Positioning System Based Routing Algorithm for Adaptive Delay Tolerant Mobile Adhoc Networks. International Journal of Computer and Communication System Engineering. 2015 May 11, 2(3): 399-406.

[19] Ananth P, La Placa RL. Secure software leasing. InAdvances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part II 2021 Jun 16 (pp. 501-530). Cham: Springer International Publishing.

[20] Hoglund G, McGraw G. Exploiting software: How to break code. Pearson Education India. 2004 Jan 1: 1-44.

[21] Hassan M, Herdt V, Le HM, Große D, Drechsler R. Early SoC security validation by VP-based static information flow analysis. In2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2017 Nov 13 (pp. 400-407). IEEE.

[22] Xu H, Heijmans J, Visser J. A practical model for rating software security. In2013 IEEE seventh international conference on software security and reliability companion 2013 Jun 18 (pp. 231-232). IEEE.

[23] Jog VV. Data importance and feedback based adaptive level of authorization for the security of Internet of Things. Journal of Engineering Research. 2019 Aug 7, 7(3).

[24] 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.

[25] 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.

[26] Samonas S, Coss D. The CIA strikes back: Redefining confidentiality, integrity and availability in security. Journal of Information System Security. 2014 Jul 1, 10(3).

[27] Tchernykh A, Schwiegelsohn U, Talbi EG, Babenko M. Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. Journal of Computational Science. 2019 Sep 1, 36:100581.

[28] Kumar R, Bhatia MP. A Systematic review of the security in cloud computing: Data integrity, confidentiality and availability. In2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON) 2020 Oct 2 (pp. 334-337). IEEE.

[29] Omollo VN, Musyoki S. Blue bugging Java Enabled Phones via Bluetooth Protocol Stack Flaws. International Journal of Computer and Communication System Engineering. 2015 Jun 9, 2 (4):608-613.

[30] Khan R. Secure software development: a prescriptive framework. Computer Fraud & Security. 2011 Aug 1, 2011(8):12-20.

[31] Meneely A, Williams L. Secure open source collaboration: an empirical study of linus' law. InProceedings of the 16th ACM conference on Computer and communications security 2009 Nov 9 (pp. 453-462).

[32] Peters R, Zaidman A. Evaluating the lifespan of code smells using software repository mining. In2012 16th European conference on software maintenance and reengineering 2012 Mar 27 (pp. 411-416). IEEE.

[33] Zimmermann T, Premraj R, Zeller A. Predicting defects for eclipse. InThird International Workshop on Predictor Models in Software Engineering (PROMISE'07: ICSE Workshops 2007) 2007 May 20 (pp. 9-9). IEEE.

[34] 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.

[35] Jiang Y, Cuki B, Menzies T, Bartlow N. Comparing design and code metrics for software quality prediction. InProceedings of the 4th international workshop on Predictor models in software engineering 2008 May 12 (pp. 11-18).

[36] Diamantopoulos T, Thomopoulos K, Symeonidis A. QualBoa: reusability-aware recommendations of source code components. InProceedings of the 13th International Conference on Mining Software Repositories 2016 May 14 (pp. 488-491).

[37] Kumar R, Pandey SK, Ahson SI. Security in coding phase of SDLC. In2007 Third International Conference on Wireless Communication and Sensor Networks 2007 Dec 13 (pp. 118-120). IEEE.

[38] Langweg H, Snekkenes E. A classification of malicious software attacks. InIEEE International Conference on Performance, Computing, and Communications, 2004 2004 Apr 15 (pp. 827-832). IEEE.

[39] Dhawan M, Poddar R, Mahajan K, Mann V. Sphinx: detecting security attacks in software-defined networks. InNdss 2015 Feb 8 (Vol. 15, pp. 8-11).

[40] Ammar M, Russello G, Crispo B. Internet of Things: A survey on the security of IoT frameworks. Journal of Information Security and Applications. 2018 Feb 1, 38:8-27.

[41] Cotroneo D, Pietrantuono R, Russo S, Trivedi K. How do bugs surface? A comprehensive study on the characteristics of software bugs manifestation. Journal of Systems and Software. 2016 Mar 1, 113:27-43.

[42] 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.

[43] Zhang M, de Carnavalet XD, Wang L, Ragab A. Large-scale empirical study of important features indicative of discovered vulnerabilities to assess application security. IEEE Transactions on Information Forensics and Security. 2019 Jan 29, 14(9):2315-30.

[44] McGraw G. Six tech trends impacting software security. Computer. 2017 May 10, 50(5):100-2.

[45] Conklin WA, Dietrich G. Secure software engineering: A new paradigm. In2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07) 2007 Jan 3 (pp. 272-272). IEEE.

[46] Mohagheghi P, Sæther T. Software engineering challenges for migration to the service cloud paradigm: Ongoing work in the remics project. In2011 IEEE World Congress on Services 2011 Jul 4 (pp. 507-514). IEEE.

[47] Sankhwar S, Singh V, Pandey D. Requirement engineering paradigm. Global Journal of Multidisciplinary Studies. 2014 Feb 3, 3(3):1-8.

[48] Núñez JC, Lindo AC, Rodríguez PG. A preventive secure software development model for a software factory: a case study. IEEE Access. 2020 Apr 21, 8:77653-65.

[49] 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.

[50] Von Solms S, Futcher LA. Adaption of a secure software development methodology for secure engineering design. IEEE Access. 2020 Jul 6, 8:125630-7.

[51] Gunduz MZ, Das R. Cyber-security on smart grid: Threats and potential solutions. Computer networks. 2020 Mar 14, 169:107094.

[52] Aguilar J, Sanchez M, Fernandez-y-Fernandez C, Rocha E, Martinez D, Figueroa J. The size of software projects developed by mexican companies. arXiv preprint arXiv:1408.1068. 2014 Aug 5.

[53] Perry DE, Siy HP, Votta LG. Parallel changes in large-scale software development: an observational case study. ACM Transactions on Software Engineering and Methodology (TOSEM). 2001 Jul 1, 10(3):308-37.

[54] Funk A, Basili V, Hochstein L, Kepner J. Application of a development time productivity metric to parallel software development. InProceedings of the second international workshop on Software engineering for high performance computing system applications 2005 May 15 (pp. 8-12).

[55] Hinsen K. High-level parallel software development with Python and BSP. Parallel Processing Letters. 2003 Sep, 13(03):473-84.

[56] Nugroho S, Waluyo SH, Hakim L. Comparative analysis of software development methods between Parallel, V-Shaped and Iterative. arXiv preprint arXiv:1710.07014. 2017 Oct 19.

[57] Mutlaq KA, Nyangaresi VO, Omar MA, Abduljabbar ZA. Symmetric Key Based Scheme for Verification Token Generation in Internet of Things Communication Environment. InApplied Cryptography in Computer and Communications: Second EAI International Conference, AC3 2022, Virtual Event, May 14-15, 2022, Proceedings 2022 Oct 6 (pp. 46-64). Cham: Springer Nature Switzerland.

[58] Jayaram KR, Mathur AP. Software engineering for secure software-state of the art: A survey. Department of Computer Science, Purdue University, USA. 2005 Sep 19.

[59] De Win B, Piessens F, Joosen W, Verhanneman T. On the importance of the separation-of-concerns principle in secure software engineering. InWorkshop on the Application of Engineering Principles to System Security Design 2002 Nov 6 (pp. 1-10).

[60] Yuan X, Yang L, Jones B, Yu H, Chu BT. Secure software engineering education: Knowledge area, curriculum and resources. Journal of Cybersecurity Education, Research and Practice. 2016, 2016(1):3.

[61] Walden J, Frank CE. Secure software engineering teaching modules. InProceedings of the 3rd annual conference on Information security curriculum development 2006 Sep 22 (pp. 19-23).

[62] Hein D, Saiedian H. Secure software engineering: Learning from the past to address future challenges. Information Security Journal: A Global Perspective. 2009 Feb 6, 18(1):8-25.

[63] Davis N, Humphrey W, Redwine ST, Zibulski G, McGraw G. Processes for producing secure software. IEEE Security & Privacy. 2004 Jun 21, 2(3):18-25.

[64] 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).

[65] Gao S, Li Z, Xiao B, Wei G. Security threats in the data plane of software-defined networks. IEEE network. 2018 Feb 7, 32(4):108-13.

[66] Shu Z, Wan J, Li D, Lin J, Vasilakos AV, Imran M. Security in software-defined networking: Threats and countermeasures. Mobile Networks and Applications. 2016 Oct, 21:764-76.

[67] Pearce M, Zeadally S, Hunt R. Virtualization: Issues, security threats, and solutions. ACM Computing Surveys (CSUR). 2013 Mar 12, 45(2):1-39.

[68] Lal S, Taleb T, Dutta A. NFV: Security threats and best practices. IEEE Communications Magazine. 2017 May 15, 55(8):211-7.

[69] Thakur K, Qiu M, Gai K, Ali ML. An investigation on cyber security threats and security models. In2015 IEEE 2nd international conference on cyber security and cloud computing 2015 Nov 3 (pp. 307-311). IEEE.

[70] Abduljabbar ZA, Nyangaresi VO, Ma J, Al Sibahee MA, Khalefa MS, Honi DG. MAC-Based Symmetric Key Protocol for Secure Traffic Forwarding in Drones. InFuture Access Enablers for Ubiquitous and Intelligent Infrastructures: 6th EAI International Conference, FABULOUS 2022, Virtual Event, May 4, 2022, Proceedings 2022 Sep 18 (pp. 16-36). Cham: Springer International Publishing.

[71] Hanif H, Nasir MH, Ab Razak MF, Firdaus A, Anuar NB. The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches. Journal of Network and Computer Applications. 2021 Apr 1, 179:103009.

[72] Ghanbari Z, Rahmani Y, Ghaffarian H, Ahmadzadegan MH. Comparative approach to web application firewalls. In2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI) 2015 Nov 5 (pp. 808-812). IEEE.

[73] Ghanbari Z, Rahmani Y, Ghaffarian H, Ahmadzadegan MH. Comparative approach to web application firewalls. In2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI) 2015 Nov 5 (pp. 808-812). IEEE.

[74] Shah S, Mehtre BM. An overview of vulnerability assessment and penetration testing techniques. Journal of Computer Virology and Hacking Techniques. 2015 Feb, 11:27-49.

[75] Al-Matouq H, Mahmood S, Alshayeb M, Niazi M. A maturity model for secure software design: A multivocal study. IEEE Access. 2020 Nov 24, 8:215758-76.

[76] Hudaib A, AlShraideh M, Surakhi O, Khanafseh M. A survey on design methods for secure software development. Int. J. Comput. Technol. 2017 Dec 10, 16(7).

[77] 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.

[78] Kannan K, Telang R. Market for software vulnerabilities? Think again. Management science. 2005 May, 51(5):726-40.

[79] Deshmukh M, Jain A. Lean-SE: Framework Combining Lean Thinking with the SDLC Process. InUbiquitous Intelligent Systems: Proceedings of ICUIS 2021 2022 (pp. 127-133). Springer Singapore.

[80] Tang J. Towards automation in software test life cycle based on multi-agent. In2010 International Conference on Computational Intelligence and Software Engineering 2010 Dec 10 (pp. 1-4). IEEE.

[81] Anthony AR, Prasad GD, Randunuge SU, Alahakoon SR, Wijendra DR, Krishara J. Software Development Automation: An Approach to Automate the Processes of SDLC. International Journal of Computer Applications. 2020, 975:8887.

[82] Garg A, Kaliyar RK, Goswami A. PDRSD-A systematic review on plan-driven SDLC models for software development. In2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS) 2022 Mar 25 (Vol. 1, pp. 739-744). IEEE.

[83] Hlaing SZ, Ochimizu K. An integrated cost-effective security requirement engineering process in SDLC using FRAM. In2018 International Conference on Computational Science and Computational Intelligence (CSCI) 2018 Dec 12 (pp. 852-857). IEEE.

[84] Jouini M, Rabai LB, Aissa AB. Classification of security threats in information systems. Procedia Computer Science. 2014 Jan 1, 32:489-96.

[85] Abduljabbar ZA, Omollo Nyangaresi V, Al Sibahee MA, Ghrabat MJ, Ma J, Qays Abduljaleel I, Aldarwish AJ. Session-Dependent Token-Based Payload Enciphering Scheme for Integrity Enhancements in Wireless Networks. Journal of Sensor and Actuator Networks. 2022 Sep 19, 11(3):55.

[86] Karim NS, Albuolayan A, Saba T, Rehman A. The practice of secure software development in SDLC: an investigation through existing model and a case study. Security and Communication Networks. 2016 Dec, 9(18):5333-45.

[87] Khari M, Kumar P. Embedding security in software development life cycle (SDLC). In2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom) 2016 Mar 16 (pp. 2182-2186). IEEE.

[88] Subedi B, Alsadoon A, Prasad PW, Elchouemi A. Secure paradigm for web application development. In2016 15th RoEduNet Conference: Networking in Education and Research 2016 Sep 7 (pp. 1-6). IEEE.

[89] Offutt J. Quality attributes of web software applications. IEEE software. 2002 Aug 7, 19(2):25-32.

[90] Thomas TW, Tabassum M, Chu B, Lipford H. Security during application development: An application security expert perspective. InProceedings of the 2018 CHI Conference on Human Factors in Computing Systems 2018 Apr 21 (pp. 1-12).

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

[92] Tiwari PK, Joshi S. Data security for software as a service. InWeb-based services: Concepts, methodologies, tools, and applications 2016 (pp. 864-880). IGI Global.

[93] Hosseinzadeh S, Rauti S, Laurén S, Mäkelä JM, Holvitie J, Hyrynsalmi S, Leppänen V. Diversification and obfuscation techniques for software security: A systematic literature review. Information and Software Technology. 2018 Dec 1, 104:72-93.

[94] Seacord RC. Secure Coding in C and C++. Pearson Education, 2005 Sep 9.

[95] Kleidermacher DN. Integrating static analysis into a secure software development process. In2008 IEEE Conference on Technologies for Homeland Security 2008 May 12 (pp. 367-371). IEEE.

[96] Shirazi HM. A new model for secure software development. International Journal of Intelligent Information Technology Application. 2009 Jun 1, 2(3).

[97] 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.

[98] Al-Shorafat WS. Security in software engineering requirement. In8th International Conference for Internet Technology and Secured Transactions (ICITST-2013) 2013 Dec 9 (pp. 666-673). IEEE.

[99] Pedraza-García G, Noël R, Matalonga S, Astudillo H, Fernandez EB. Mitigating security threats using tactics and patterns: a controlled experiment. InProccedings of the 10th European Conference on Software Architecture Workshops 2016 Nov 28 (pp. 1-7).

[100] Wu J, Zou L, Zhao L, Al-Dubai A, Mackenzie L, Min G. A multi-UAV clustering strategy for reducing insecure communication range. Computer Networks. 2019 Jul 20, 158:132-42.

[101] 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.

[102] Hazeyama A, Saito M, Yoshioka N, Kumagai A, Kobashi T, Washizaki H, Kaiya H, Okubo T. Case base for secure software development using software security knowledge base. In2015 IEEE 39th Annual Computer Software and Applications Conference 2015 Jul 1 (Vol. 3, pp. 97-103). IEEE.

[103] Sharieh S, Ferworn A. Securing APIs and Chaos Engineering. In2021 IEEE Conference on Communications and Network Security (CNS) 2021 Oct 4 (pp. 290-294). IEEE.

[104] Conti M, Dragoni N, Lesyk V. A survey of man in the middle attacks. IEEE communications surveys & tutorials. 2016 Mar 29, 18(3):2027-51.

[105] Braz L, Fregnan E, Çalikli G, Bacchelli A. Why Don't Developers Detect Improper Input Validation?', DROP TABLE Papers, --. In2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) 2021 May 22 (pp. 499-511). IEEE.

[106] Al Sibahee MA, Nyangaresi VO, Ma J, Abduljabbar ZA. Stochastic Security Ephemeral Generation Protocol for 5G Enabled Internet of Things. InIoT as a Service: 7th EAI International Conference, IoTaaS 2021, Sydney, Australia, December 13–14, 2021, Proceedings 2022 Jul 8 (pp. 3-18). Cham: Springer International Publishing.

[107] Ghasemisharif M, Ramesh A, Checkoway S, Kanich C, Polakis J. O Single Sign-Off, Where Art Thou? An Empirical Analysis of Single Sign-On Account Hijacking and Session Management on the Web. InUSENIX Security Symposium 2018 Aug 15 (pp. 1475-1492).

[108] Deepa G, Thilagam PS. Securing web applications from injection and logic vulnerabilities: Approaches and challenges. Information and Software Technology. 2016 Jun 1, 74:160-80.

[109] Li L, Correia PL, Hadid A. Face recognition under spoofing attacks: countermeasures and research directions. Iet Biometrics. 2018 Jan, 7(1):3-14.

[110] Kremer S, Markowitch O, Zhou J. An intensive survey of fair non-repudiation protocols. Computer communications. 2002 Nov 1, 25(17):1606-21.

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

[112] Mufti Y, Niazi M, Alshayeb M, Mahmood S. A readiness model for security requirements engineering. IEEE Access. 2018 May 24, 6:28611-31.

[113] Banowosari LY, Gifari BA. System analysis and design using secure software development life cycle based on ISO 31000 and STRIDE. Case study mutiara ban workshop. In2019 Fourth International Conference on Informatics and Computing (ICIC) 2019 Oct 16 (pp. 1-6). IEEE.

[114] Pedraza-Garcia G, Astudillo H, Correal D. A methodological approach to apply security tactics in software architecture design. In2014 IEEE Colombian Conference on Communications and Computing (COLCOM) 2014 Jun 4 (pp. 1-8). IEEE.

[115] Apvrille A, Pourzandi M. Secure software development by example. IEEE Security & Privacy. 2005 Aug 8, 3(4):10-7.

[116] Trnka M, Cerny T, Stickney N. Survey of Authentication and Authorization for the Internet of Things. Security and Communication Networks. 2018 Jun 12, 2018.

[117] Nyangaresi VO, Abduljabbar ZA, Mutlaq KA, Hussain MA, Hussien ZA. Forward and Backward Key Secrecy Preservation Scheme for Medical Internet of Things. InHuman-Centric Smart Computing: Proceedings of ICHCSC 2022 2022 Nov 29 (pp. 15-29). Singapore: Springer Nature Singapore.

[118] Cope R. Strong security starts with software development. Network Security. 2020 Jul, 2020(7):6-9.

[119] Kaur D, Kaur P. Empirical analysis of web attacks. Procedia Computer Science. 2016 Jan 1, 78:298-306.

[120] Palsetia N, Deepa G, Khan FA, Thilagam PS, Pais AR. Securing native XML database-driven web applications from XQuery injection vulnerabilities. Journal of Systems and Software. 2016 Dec 1, 122:93-109.

[121] Ron A, Shulman-Peleg A, Bronshtein E. No sql, no injection? examining nosql security. arXiv preprint arXiv:1506.04082. 2015 Jun 12.

[122] Ron A, Shulman-Peleg A, Puzanov A. Analysis and mitigation of NoSQL injections. IEEE Security & Privacy. 2016 Apr 6, 14(2):30-9.

[123] Junjin M. An approach for SQL injection vulnerability detection. In2009 Sixth international conference on information technology: new generations 2009 Apr 27 (pp. 1411-1414). IEEE.

[124] 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.

[125] Liang G, Zhao J, Luo F, Weller SR, Dong ZY. A review of false data injection attacks against modern power systems. IEEE Transactions on Smart Grid. 2016 Mar 22, 8(4):1630-8.

[126] Almorsy M, Grundy J, Ibrahim AS. Automated software architecture security risk analysis using formalized signatures. In2013 35th International Conference on Software Engineering (ICSE) 2013 May 18 (pp. 662-671). IEEE.

[127] Yu Y, Xiao G, Zhou J, Wang Y, Wang Z, Kurths J, Schellnhuber HJ. System crash as dynamics of complex networks. Proceedings of the National Academy of Sciences. 2016 Oct 18, 113(42):11726-31.

[128] Bornholt J, Kaufmann A, Li J, Krishnamurthy A, Torlak E, Wang X. Specifying and checking file system crash-consistency models. InProceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems 2016 Mar 25 (pp. 83-98).

[129] Khan MU, Zulkernine M. On selecting appropriate development processes and requirements engineering methods for secure software. In2009 33rd Annual IEEE International Computer Software and Applications Conference 2009 Jul 20 (Vol. 2, pp. 353-358). IEEE.

[130] ur Rehman I, Malik SU. The impact of test case reduction and prioritization on software testing effectiveness. In2009 International Conference on Emerging Technologies 2009 Oct 19 (pp. 416-421). IEEE.

[131] 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.

[132] Mouratidis H, Giorgini P, Manson G. When security meets software engineering: a case of modelling secure information systems. Information Systems. 2005 Dec 1, 30(8):609-29.

[133] Goertzel KM. Legal liability for bad software. CrossTalk. 2016 Sep, 23.

[134] Kim S. Crashed software: Assessing product liability for software defects in automated vehicles. Duke L. & Tech. Rev.. 2017, 16:300.

[135] Scott MD. Tort liability for vendors of insecure software: Has the time finally come. Md. L. Rev.. 2007, 67:425.

[136] Farhan AS, Mostafa GM. A methodology for enhancing software security during development processes. In2018 21st Saudi Computer Society National Computer Conference (NCC) 2018 Apr 25 (pp. 1-6). IEEE.

[137] Avoine G, Carpent X, Hernandez-Castro J. Pitfalls in ultralightweight authentication protocol designs. IEEE Transactions on Mobile Computing. 2015 Oct 19, 15(9):2317-32.

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

[139] McLeod A, Dolezel D. Information security policy non-compliance: Can capitulation theory explain user behaviors?. Computers & Security. 2022 Jan 1, 112:102526.

[140] Asad M, Ahmed S. Model driven architecture for secure software development life cycle. International Journal of Computer Science and Information Security (IJCSIS). 2016 Jun, 14(6).

[141] Velmourougan S, Dhavachelvan P, Baskaran R, Ravikumar B. Software development life cycle model to improve maintainability of software applications. In2014 Fourth International Conference on Advances in Computing and Communications 2014 Aug 27 (pp. 270-273). IEEE.

[142] Catuogno L, Galdi C, Persiano G. Secure dependency enforcement in package management systems. IEEE Transactions on Dependable and Secure Computing. 2017 Nov 27, 17(2):377-90.

[143] Basin D, Doser J, Lodderstedt T. Model driven security: From UML models to access control infrastructures. ACM Transactions on Software Engineering and Methodology (TOSEM). 2006 Jan 1, 15(1):39-91.

[144] 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.

[145] Xiong W, Lagerström R. Threat modeling–A systematic literature review. Computers & security. 2019 Jul 1, 84:53-69.

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

[147] Yilmaz M, O'Connor RV. A Scrumban integrated gamification approach to guide software process improvement: a Turkish case study. Tehnički vjesnik. 2016 Feb 19, 23(1):237-45.

[148] Marback A, Do H, He K, Kondamarri S, Xu D. A threat model-based approach to security testing. Software: Practice and Experience. 2013 Feb, 43(2):241-58.

[149] Tung YH, Lo SC, Shih JF, Lin HF. An integrated security testing framework for secure software development life cycle. In2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS) 2016 Oct 5 (pp. 1-4). IEEE.

[150] 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.

[151] 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.

[152] Mead NR. Measuring the software security requirements engineering process. In2012 IEEE 36th Annual Computer Software and Applications Conference Workshops 2012 Jul 16 (pp. 583-588). IEEE.

[153] Kumar C, Yadav DK. Software defects estimation using metrics of early phases of software development life cycle. International Journal of System Assurance Engineering and Management. 2017 Dec, 8:2109-17.

[154] Yadav HB, Yadav DK. A fuzzy logic based approach for phase-wise software defects prediction using software metrics. Information and Software Technology. 2015 Jul 1, 63:44-57.

[155] Aljawarneh SA, Alawneh A, Jaradat R. Cloud security engineering: Early stages of SDLC. Future Generation Computer Systems. 2017 Sep 1, 74:385-92.

[156] 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).

[157] Banerjee C, Banerjee A, Murarka PD. Evaluating the relevance of prevailing software metrics to address issue of security implementation in SDLC. International Journal of Advanced Studies in Computers, Science and Engineering. 2014 Mar 1, 3(3):18.

[158] Yahya S, Kamalrudin M, Sidek S, Jaimun M, Yusof J, Hua AK, Gani P. A review paper: Security requirement patterns for a secure software development. In2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS) 2019 Sep 19 (pp. 146-151). IEEE.

[159] Salini P, Kanmani S. Survey and analysis on security requirements engineering. Computers & Electrical Engineering. 2012 Nov 1, 38(6):1785-97.

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

[161] Van den Berghe A, Scandariato R, Yskout K, Joosen W. Design notations for secure software: a systematic literature review. Software & Systems Modeling. 2017 Jul, 16:809-31.

[162] Maheshwari V, Prasanna M. Integrating risk assessment and threat modeling within SDLC process. In2016 international conference on inventive computation technologies (ICICT) 2016 Aug 26 (Vol. 1, pp. 1-5). IEEE.

[163] Khan K, Ahmad R, Yazid I. 'Systematic mapping study protocol for secure software engineering. Proc. AIMC. 2019:367-74.

[164] Doan T, Demurjian S, Ting TC, Ketterl A. MAC and UML for secure software design. InProceedings of the 2004 ACM workshop on Formal Methods in Security Engineering 2004 Oct 29 (pp. 75-85).

[165] Khan RA, Khan SU. A preliminary structure of software security assurance model. InProceedings of the 13th International Conference on Global Software Engineering 2018 May 27 (pp. 137-140).

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

[167] Mousa A, Karabatak M, Mustafa T. Database security threats and challenges. In2020 8th International Symposium on Digital Forensics and Security (ISDFS) 2020 Jun 1 (pp. 1-5). IEEE.

[168] Bird C, Menzies T, Zimmermann T, editors. The art and science of analyzing software data. Elsevier, 2015 Sep 2.

[169] Dilawer SA. Practical Guide of Software Development Project Management in Practice. Lulu. com, 2011.

[170] Mack H, Schroer T. Security midlife crisis: Building security in a new world. IEEE Security & Privacy. 2020 Jul 10, 18(4):72-4.

[171] Pittenger M. Open source security analysis: The state of open source security in commercial applications. Black Duck Software, Tech. Rep. 2016.

[172] Gallivan MJ. Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies. Information Systems Journal. 2001 Oct, 11(4):277-304.

[173] Cox J, Bouwers E, Van Eekelen M, Visser J. Measuring dependency freshness in software systems. In2015 IEEE/ACM 37th IEEE International Conference on Software Engineering 2015 May 16 (Vol. 2, pp. 109-118). IEEE.

[174] Pashchenko I, Plate H, Ponta SE, Sabetta A, Massacci F. Vulnerable open source dependencies: Counting those that matter. InProceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement 2018 Oct 11 (pp. 1-10).

[175] Pashchenko I, Vu DL, Massacci F. A qualitative study of dependency management and its security implications. InProceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security 2020 Oct 30 (pp. 1513-1531).

[176] 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.

[177] Kula RG, German DM, Ouni A, Ishio T, Inoue K. Do developers update their library dependencies? An empirical study on the impact of security advisories on library migration. Empirical Software Engineering. 2018 Feb, 23:384-417.

[178] Dashevskyi S, Brucker AD, Massacci F. On the effort for security maintenance of free and open source components. InProc. of WEIS 2018 (Vol. 18).

[179] Derr E, Bugiel S, Fahl S, Acar Y, Backes M. Keep me updated: An empirical study of third-party library updatability on android. InProceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security 2017 Oct 30 (pp. 2187-2200).

[180] Hejderup J. In Dependencies We Trust: How vulnerable are dependencies in software modules? Master's thesis. Delft University of Technology, Delft, the Netherlands. 2015.

[181] Lauinger T, Chaabane A, Arshad S, Robertson W, Wilson C, Kirda E. Thou shalt not depend on me: Analysing the use of outdated javascript libraries on the web. arXiv preprint arXiv:1811.00918. 2018 Nov 2.

[182] Abduljabbar ZA, Abduljaleel IQ, Ma J, Al Sibahee MA, Nyangaresi VO, Honi DG, Abdulsada AI, Jiao X. Provably secure and fast color image encryption algorithm based on s-boxes and hyperchaotic map. IEEE Access. 2022 Feb 11, 10:26257-70.

[183] Li J, Li YK, Chen X, Lee PP, Lou W. A hybrid cloud approach for secure authorized deduplication. IEEE transactions on parallel and distributed systems. 2014 Apr 18, 26(5):1206-16.

[184] Al Sibahee MA, Abdulsada AI, Abduljabbar ZA, Ma J, Nyangaresi VO, Umran SM. Lightweight, Secure, Similar-Document Retrieval over Encrypted Data. Applied Sciences. 2021 Jan, 11(24):12040.

[185] Sabale RG, Dani AR. Comparative study of prototype model for software engineering with system development life cycle. IOSR Journal of Engineering. 2012 Jul, 2(7):21-4.

[186] Assal H, Chiasson S. Security in the Software Development Lifecycle. InSOUPS@ USENIX Security Symposium 2018 Aug 12 (pp. 281-296).

[187] Eian IC, Yong LK, Li MY, Hasmaddi NA. Integration of Security Modules in Software Development Lifecycle Phases. arXiv preprint arXiv:2012.05540. 2020 Dec 10.

[188] 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.

[189] 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.

[190] Essafi M, Labed L, Ghezala HB. S2D-Prom: A strategy oriented process model for secure software development. InInternational Conference on Software Engineering Advances (ICSEA 2007) 2007 Aug 25 (pp. 24-24). IEEE.

[191] Syed R, Rahafrooz M, Keisler JM. What it takes to get retweeted: An analysis of software vulnerability messages. Computers in Human Behavior. 2018 Mar 1, 80:207-15.

[192] Giffin JT, Christodorescu M, Kruger L. Strengthening software self-checksumming via self-modifying code. In21st Annual Computer Security Applications Conference (ACSAC'05) 2005 Dec 5 (pp. 10-pp). IEEE.

[193] Moe NB, Šmite D. Understanding a lack of trust in Global Software Teams: a multiple-case study. Software Process: Improvement and Practice. 2008 May, 13(3):217-31.

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

[195] Maher ZA, Shaikh H, Khan MS, Arbaaeen A, Shah A. Factors affecting secure software development practices among developers-An investigation. In2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS) 2018 Nov 22 (pp. 1-6). IEEE.

[196] Rodríguez GE, Torres JG, Flores P, Benavides DE. Cross-site scripting (XSS) attacks and mitigation: A survey. Computer Networks. 2020 Jan 15, 166:106960.

[197] 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.

[198] Silva P, Noël R, Gallego M, Matalonga S, Astudillo H. Software Development Initiatives to Identify and Mitigate Security Threats: A Systematic Mapping. InCIbSE 2016 Apr 27 (pp. 257-270).

[199] Guinea AS, Nain G, Le Traon Y. A systematic review on the engineering of software for ubiquitous systems. Journal of Systems and Software. 2016 Aug 1, 118:251-76.

[200] Rafi S, Yu W, Akbar MA, Alsanad A, Gumaei A. Prioritization based taxonomy of DevOps security challenges using PROMETHEE. IEEE Access. 2020 Jun 1, 8:105426-46.

[201] Hall A, Chapman R. Correctness by construction: Developing a commercial secure system. IEEE software. 2002 Aug 7, 19(1):18-25.

[202] 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.

[203] Potter B, McGraw G. Software security testing. IEEE Security & Privacy. 2004 Oct 8, 2(5):81-5.

[204] Verdon D, McGraw G. Risk analysis in software design. IEEE Security & Privacy. 2004 Oct 4, 2(4):79-84.

[205] Lipner S. The trustworthy computing security development lifecycle. In20th Annual Computer Security Applications Conference 2004 Dec 6 (pp. 2-13). IEEE.

[206] Gupta S, Faisal M, Husain M. Secure software development process for embedded systems control. Int. J. Eng. Sci. Emerg. Technol. 2012 Dec 1, 4:133-43.

[207] 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.

[208] Manico J. OWASP. Proc. Appl. Secur. Verification Standard. 2016:1-70.

[209] Gregoire J, Buyens K, De Win B, Scandariato R, Joosen W. On the secure software development process: CLASP and SDL compared. InThird International Workshop on Software Engineering for Secure Systems (SESS'07: ICSE Workshops 2007) 2007 May 20 (pp. 1-1). IEEE.

[210] Li W, Chiueh TC. Automated format string attack prevention for win32/x86 binaries. InTwenty-Third Annual Computer Security Applications Conference (ACSAC 2007) 2007 Dec 10 (pp. 398-409). IEEE.

[211] Sahu DR, Tomar DS. Analysis of web application code vulnerabilities using secure coding standards. Arabian Journal for Science and Engineering. 2017 Feb, 42:885-95.

[212] 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.

[213] Gasiba TE, Lechner U, Pinto-Albuquerque M, Mendez D. Is secure coding education in the industry needed? An investigation through a large scale survey. In2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET) 2021 May 25 (pp. 241-252). IEEE.

[214] 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.

[215] Peine H. Rules of thumb for developing secure software: Analyzing and consolidating two proposed sets of rules. In2008 Third International Conference on Availability, Reliability and Security 2008 Mar 4 (pp. 1204-1209). IEEE.

[216] Mead NR, Stehney T. Security quality requirements engineering (SQUARE) methodology. ACM SIGSOFT Software Engineering Notes. 2005 May 15, 30(4):1-7.

[217] Flechais I, Mascolo C, Sasse MA. Integrating security and usability into the requirements and design process. International Journal of Electronic Security and Digital Forensics. 2007 Jan 1, 1(1):12-26.

[218] Collins B. Big data and health economics: strengths, weaknesses, opportunities and threats. Pharmacoeconomics. 2016 Feb, 34(2):101-6.

[219] Sodiya AS, Onashoga SA, Ajayĩ OB. Towards building secure software systems. Issues in Informing Science & Information Technology. 2006 Jan 1, 3.

[220] Hentea M, Dhillon HS, Dhillon M. Towards changes in information security education. Journal of Information Technology Education: Research. 2006 Jan 1, 5(1):221-33.

[221] 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.

[222] Aslanyan Z, Nielson F, Parker D. Quantitative verification and synthesis of attack-defence scenarios. In2016 IEEE 29th Computer Security Foundations Symposium (CSF) 2016 Jun 27 (pp. 105-119). IEEE.

[223] Nguyen TH, Grundy J, Almorsy M. Integrating goal-oriented and use case-based requirements engineering: The missing link. In2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS) 2015 Sep 30 (pp. 328-337). IEEE.

[224] Vassilakis VG, Mouratidis H, Panaousis E, Moscholios ID, Logothetis MD. Security requirements modelling for virtualized 5G small cell networks. In2017 24th International Conference on Telecommunications (ICT) 2017 May 3 (pp. 1-5). IEEE.

[225] Fernandez-Buglioni E. Security patterns in practice: designing secure architectures using software patterns. John Wiley & Sons, 2013 Jun 25.

[226] Xu D, Nygard KE. Threat-driven modeling and verification of secure software using aspect-oriented Petri nets. IEEE transactions on software engineering. 2006 May 8, 32(4):265-78.

[227] Mellado D, Fernández-Medina E, Piattini M. A common criteria based security requirements engineering process for the development of secure information systems. Computer standards & interfaces. 2007 Feb 1, 29(2):244-53.

[228] Fabian B, Gürses S, Heisel M, Santen T, Schmidt H. A comparison of security requirements engineering methods. Requirements engineering. 2010 Mar, 15:7-40.

[229] Haley C, Laney R, Moffett J, Nuseibeh B. Security requirements engineering: A framework for representation and analysis. IEEE Transactions on Software Engineering. 2008 Jan 31, 34(1):133-53.

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

[231] Humphreys T. State-of-the-art information security management systems with ISO/IEC 27001: 2005. ISO Management Systems. 2006 Jan, 6(1):15-8.

[232] Mataracioglu T, Ozkan S. Analysis of the user acceptance for implementing ISO/IEC 27001: 2005 in turkish public organizations. arXiv preprint arXiv:1103.0405. 2011 Mar 2.

[233] Costin D, Militaru C. Asset Management Towards ISO/IEC 27001: 2005 Accreditation of an Information Security Management System. Revista De Management Comparat International/Review of International Comparative Management. 2011, 12(6):245-50.

[234] Soomro ZA, Shah MH, Ahmed J. Information security management needs more holistic approach: A literature review. International journal of information management. 2016 Apr 1, 36(2):215-25.

[235] Zhao Y, Gu P, Zhu F, Liu T, Shen R. Security control scheme for cyber-physical system with a complex network in physical layer against false data injection attacks. Applied Mathematics and Computation. 2023 Jun 15, 447:127908.

[236] Nyangaresi VO, Mohammad Z. Session Key Agreement Protocol for Secure D2D Communication. InThe Fifth International Conference on Safety and Security with IoT: SaSeIoT 2021 2022 Jun 12 (pp. 81-99). Cham: Springer International Publishing.

[237] Thompson C, Shelton M, Stark E, Walker M, Schechter E, Felt AP. The web's identity crisis: understanding the effectiveness of website identity indicators. In28th {USENIX} Security Symposium ({USENIX} Security 19) 2019 (pp. 1715-1732).

[238] Younas M, Jawawi DN, Shah MA, Mustafa A, Awais M, Ishfaq MK, Wakil K. Elicitation of nonfunctional requirements in agile development using cloud computing environment. IEEE access. 2020 Aug 27, 8:209153-62.

[239] Preuveneers D, Berbers Y, Bhatti G. Best practices for software security: An overview. In2008 IEEE International Multitopic Conference 2008 Dec 23 (pp. 169-173). IEEE.

[240] Khan RA, Khan SU, Ilyas M, Idris MY. The state of the art on secure software engineering: A systematic mapping study. Proceedings of the Evaluation and Assessment in Software Engineering. 2020 Apr 15:487-92.

[241] 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.

[242] ben Othmane 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.

[243] Mumtaz H, Alshayeb M, Mahmood S, Niazi M. An empirical study to improve software security through the application of code refactoring. Information and Software Technology. 2018 Apr 1, 96:112-25.

[244] Venson E, Guo X, Yan Z, Boehm B. Costing secure software development: A systematic mapping study. InProceedings of the 14th International Conference on Availability, Reliability and Security 2019 Aug 26 (pp. 1-11).

[245] Sodanil M, Quirchmayr G, Porrawatpreyakorn N, Tjoa AM. A knowledge transfer framework for secure coding practices. In2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE) 2015 Jul 22 (pp. 120-125). IEEE.

[246] Venson E, Alfayez R, Gomes MM, Figueiredo RM, Boehm B. The impact of software security practices on development effort: An initial survey. In2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) 2019 Sep 19 (pp. 1-12). IEEE.

[247] Nyangaresi VO. A Formally Verified Authentication Scheme for mmWave Heterogeneous Networks. Inthe 6th International Conference on Combinatorics, Cryptography, Computer Science and Computation (605-612) 2021.

[248] Salini P, Kanmani S. Effectiveness and performance analysis of model-oriented security requirements engineering to elicit security requirements: a systematic solution for developing secure software systems. International Journal of Information Security. 2016 Jun, 15:319-34.

[249] Musa Shuaibu B, Md Norwawi N, Selamat MH, Al-Alwani A. Systematic review of web application security development model. Artificial Intelligence Review. 2015 Feb, 43:259-76.

[250] Grieco G, Grinblat GL, Uzal L, Rawat S, Feist J, Mounier L. Toward large-scale vulnerability discovery using machine learning. InProceedings of the Sixth ACM Conference on Data and Application Security and Privacy 2016 Mar 9 (pp. 85-96).

[251] Cui L, Cui J, Hao Z, Li L, Ding Z, Liu Y. An empirical study of vulnerability discovery methods over the past ten years. Computers & Security. 2022 Sep 1, 120:102817.

[252] Nyangaresi VO. Target Tracking Area Selection and Handover Security in Cellular Networks: A Machine Learning Approach. InProceedings of Third International Conference on Sustainable Expert Systems: ICSES 2022 2023 Feb 23 (pp. 797-816). Singapore: Springer Nature Singapore.

[253] De Win B, Scandariato R, Buyens K, Grégoire J, Joosen W. On the secure software development process: CLASP, SDL and Touchpoints compared. Information and software technology. 2009 Jul 1, 51(7):1152-71.

[254] Siddiqui ST. Significance of security metrics in secure software development. Significance. 2017 Aug, 12(6).

[255] Chess B, Arkin B. Software security in practice. IEEE Security & Privacy. 2011 Mar 28, 9(2):89-92.

[256] Al-Amin S, Ajmeri N, Du H, Berglund EZ, Singh MP. Toward effective adoption of secure software development practices. Simulation Modelling Practice and Theory. 2018 Jun 1, 85:33-46.

[257] Souag A, Salinesi C, Wattiau I, Mouratidis H. Using security and domain ontologies for security requirements analysis. In2013 IEEE 37th Annual Computer Software and Applications Conference Workshops 2013 Jul 22 (pp. 101-107). IEEE.

[258] Nyangaresi VO, Abduljabbar ZA, Ma J, Al Sibahee MA. Verifiable Security and Privacy Provisioning Protocol for High Reliability in Smart Healthcare Communication Environment. In2022 4th Global Power, Energy and Communication Conference (GPECOM) 2022 Jun 14 (pp. 569-574). IEEE.

[259] Mayvan BB, Rasoolzadegan A, Yazdi ZG. The state of the art on design patterns: A systematic mapping of the literature. Journal of Systems and Software. 2017 Mar 1, 125:93-118.

[260] Kadam SP, Joshi S. Secure by design approach to improve security of object oriented software. In2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom) 2015 Mar 11 (pp. 24-30). IEEE.

[261] Janakiraman S, Thenmozhi K, Rayappan JB, Amirtharajan R. Lightweight chaotic image encryption algorithm for real-time embedded system: Implementation and analysis on 32-bit microcontroller. Microprocessors and Microsystems. 2018 Feb 1, 56:1-2.

[262] Zaki Rashed 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 layer fibre with higher dominant modes using OptiFibre simulation. Journal of Optical Communications. 2022 Jun 23(0).

[263] Clegg BS, Rojas JM, Fraser G. Teaching software testing concepts using a mutation testing game. In2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET) 2017 May 20 (pp. 33-36). IEEE.

[264] Xie J, Lipford HR, Chu B. Why do programmers make security errors?. In2011 IEEE symposium on visual languages and human-centric computing (VL/HCC) 2011 Sep 18 (pp. 161-164). IEEE.

[265] Nyangaresi VO, Ma J, Al Sibahee MA, Abduljabbar ZA. Packet Replays Prevention Protocol for Secure B5G Networks. InProceedings of Seventh International Congress on Information and Communication Technology: ICICT 2022, London, Volume 2 2022 Jul 27 (pp. 507-522). Singapore: Springer Nature Singapore.

[266] Felderer M, Fourneret E. A systematic classification of security regression testing approaches. International Journal on Software Tools for Technology Transfer. 2015 Jun, 17:305-19.

[267] Arcangeli JP, Boujbel R, Leriche S. Automatic deployment of distributed software systems: Definitions and state of the art. Journal of Systems and Software. 2015 May 1, 103:198-218.

[268] Colomo-Palacios R, Fernandes E, Soto-Acosta P, Larrucea X. A case analysis of enabling continuous software deployment through knowledge management. International Journal of Information Management. 2018 Jun 1, 40:186-9.

[269] Sawadogo AD, Bissyandé TF, Moha N, Allix K, Klein J, Li L, Traon YL. Learning to catch security patches. arXiv preprint arXiv:2001.09148. 2020 Jan 24.

[270] Shi Y, Zhang Y, Luo T, Mao X, Cao Y, Wang Z, Zhao Y, Huang Z, Yang M. Backporting Security Patches of Web Applications: A Prototype Design and Implementation on Injection Vulnerability Patches. In31st USENIX Security Symposium (USENIX Security 22) 2022 (pp. 1993-2010).

[271] Midha V, Bhattacherjee A. Governance practices and software maintenance: A study of open source projects. Decision Support Systems. 2012 Dec 1, 54(1):23-32.

[272] Ghrabat MJ, Hussien ZA, Khalefa MS, Abduljabba ZA, Nyangaresi VO, Al Sibahee MA, Abood EW. Fully automated model on breast cancer classification using deep learning classifiers. Indonesian Journal of Electrical Engineering and Computer Science. 2022 Oct, 28(1):183-91.

[273] Uzunov AV, Fernandez EB, Falkner K. Engineering Security into Distributed Systems: A Survey of Methodologies. Journal of Universal Computer Science. 2012, 18(20):2920-3006.

[274] Yaseen M, Baseer S, Sherin S. Critical challenges for requirement implementation in context of global software development: A systematic literature review. In2015 International Conference on Open Source Systems & Technologies (ICOSST) 2015 Dec 17 (pp. 120-125). IEEE.

[275] Han X, Zhang J. A combined analysis method of FMEA and FTA for improving the safety analysis quality of safety-critical software. In2013 IEEE International Conference on Granular Computing (GrC) 2013 Dec 13 (pp. 353-356). IEEE.

[276] Alam M, Seifert JP, Zhang X. A model-driven framework for trusted computing based systems. In11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007) 2007 Oct 15 (pp. 75-75). IEEE.

[277] Nyangaresi VO, Al Sibahee MA, Abduljabbar ZA, Ma J, Khalefa MS. Biometric-Based Packet Validation Scheme for Body Area Network Smart Healthcare Devices. In2022 IEEE 21st Mediterranean Electrotechnical Conference (MELECON) 2022 Jun 14 (pp. 726-731). IEEE.

[278] Kim SK. Design of enhanced software protection architecture by using theory of inventive problem solving. In2009 IEEE International Conference on Industrial Engineering and Engineering Management 2009 Dec 8 (pp. 978-982). IEEE.

[279] Boucharas V, Jansen S, Brinkkemper S. Formalizing software ecosystem modeling. InProceedings of the 1st international workshop on Open component ecosystems 2009 Aug 24 (pp. 41-50).

[280] Bonaccorsi A, Rossi C. Why open source software can succeed. Research policy. 2003 Jul 1, 32(7):1243-58.

[281] Velásquez I, Caro A, Rodríguez A. Authentication schemes and methods: A systematic literature review. Information and Software Technology. 2018 Feb 1, 94:30-7.

[282] Lee Y, Lee G. HW-CDI: Hard-wired control data integrity. IEEE Access. 2019 Jan 10, 7:10811-22.

[283] Nyangaresi VO, Abduljabbar ZA, Ma J, Al Sibahee MA. Temporary Symmetric Key Based Message Verification Protocol for Smart Energy Networks. In2022 IEEE 7th International Energy Conference (ENERGYCON) 2022 May 9 (pp. 1-6). IEEE.

[284] Li J, Zhang Y, Chen X, Xiang Y. Secure attribute-based data sharing for resource-limited users in cloud computing. Computers & Security. 2018 Jan 1, 72:1-2.

[285] Sharma A, Misra PK. Aspects of enhancing security in software development life cycle. Advances in Computational Sciences and Technology. 2017, 10(2):203-10.

[286] Khreich W, Murtaza SS, Hamou-Lhadj A, Talhi C. Combining heterogeneous anomaly detectors for improved software security. Journal of Systems and Software. 2018 Mar 1, 137:415-29.

[287] Srivastava AK, Kumar S. An effective computational technique for taxonomic position of security vulnerability in software development. Journal of Computational Science. 2018 Mar 1, 25:388-96.

[288] Mellado D, Blanco C, Sánchez LE, Fernández-Medina E. A systematic review of security requirements engineering. Computer Standards & Interfaces. 2010 Jun 1, 32(4):153-65.

[289] Nyangaresi VO, Abduljabbar ZA, Refish SH, Al Sibahee MA, Abood EW, Lu S. Anonymous Key Agreement and Mutual Authentication Protocol for Smart Grids. InCognitive Radio Oriented Wireless Networks and Wireless Internet: 16th EAI International Conference, CROWNCOM 2021, Virtual Event, December 11, 2021, and 14th EAI International Conference, WiCON 2021, Virtual Event, November 9, 2021, Proceedings 2022 Mar 31 (pp. 325-340). Cham: Springer International Publishing.

[290] Islam S, Dong W. Human factors in software security risk management. InProceedings of the first international workshop on Leadership and management in software architecture 2008 May 11 (pp. 13-16).

[291] Bracciale L, Loreti P, Detti A, Paolillo R, Melazzi NB. Lightweight named object: An ICN-based abstraction for IoT device programming and management. IEEE Internet of Things Journal. 2019 Jan 23, 6(3):5029-39.

[292] Sulayman M, Mendes E. A systematic literature review of software process improvement in small and medium web companies. InAdvances in Software Engineering: International Conference on Advanced Software Engineering and Its Applications, ASEA 2009 Held as Part of the Future Generation Information Technology Conference, FGIT 2009, Jeju Island, Korea, December 10-12, 2009. Proceedings 2009 (pp. 1-8). Springer Berlin Heidelberg.

[293] Herath HS, Herath TC. IT security auditing: A performance evaluation decision model. Decision Support Systems. 2014 Jan 1, 57:54-63.

[294] Steinbart PJ, Raschke RL, Gal G, Dilla WN. The relationship between internal audit and information security: An exploratory investigation. International Journal of Accounting Information Systems. 2012 Sep 1, 13(3):228-43.

[295] 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.

[296] Heitmeyer C, Archer M, Leonard E, McLean J. Applying formal methods to a certifiably secure software system. IEEE Transactions on Software Engineering. 2008 Jan 31, 34(1):82-98.

[297] Huang J, Borges N, Bugiel S, Backes M. Up-to-crash: Evaluating third-party library updatability on android. In2019 IEEE European Symposium on Security and Privacy (EuroS&P) 2019 Jun 17 (pp. 15-30). IEEE.

[298] Choudhary D, Kumar V. Software testing. Journal of Computational Simulation and Modeling. 2011 Jan 1, 1(1):1.