Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 8 No. 1 (2024) 88 - 99 e-ISSN: 2580-0760

A Novel Framework for Information Security During the SDLC Implementation Stage: A Systematic Literature Review

Mikael Octavinus Chan[1], Setiadi Yazid[2]
[1]Information Technology, Computer Science, Universitas Indonesia, Jakarta, Indonesia
[2]Information Technology, Compter Science, Universitas Indonesia, Jakarta, Indonesia
[1]m.octavinus@ui.ac.id, [2]setiadi@cs.ui.ac.id

Abstract

This research delves into the critical aspects of information security during the implementation stage of the Software Development Life Cycle (SDLC). Using a systematic review of the literature, the study synthesizes the findings of various digital repositories, including IEEE Xplore, ACM Digital Library, Scopus, and ScienceDirect, to outline a comprehensive framework that addresses the unique security challenges of the implementation stage. This research contributes to the field by proposing a novel assurance model for software development vendors, focusing on improving information security measures during the implementation stage. The study's findings reveal 12 key steps organizations can adopt to mitigate security risks and improve information security measures during this critical phase. These steps provide actionable insights and strategies designed to support security protocols effectively. The paper concludes that by incorporating these steps, organizations can significantly improve their security posture, ensuring the integrity and reliability of the software development process, particularly during the implementation stage. This approach not only addresses immediate security concerns but also sets a precedent for future research and practice in secure software development, particularly in the critical implementation stage of the SDLC.

Keywords: information security; implementation; system development life cycle (SDLC); secure software development life cycle (SSDLC)

How to Cite: Mikael Octavinus Chan and Setiadi Yazid, "A Novel Framework for Information Security During the SDLC Implementation Stage: A Systematic Literature Review", J. RESTI (Rekayasa Sist. Teknol. Inf.) , vol. 8, no. 1, pp. 88 - 99, Feb. 2024.
DOI: https://doi.org/10.29207/resti.v8i1.5403

1. Introduction

Numerous academic works have been published to address information security within the system development life cycle (SDLC), encompassing a broad spectrum of phases. The demand for security in software development has led to the establishment of what is known as the Secure Software Development Life Cycle (SSDLC) [1]. The SSDLC places a strong emphasis on integrating security throughout the entire Software Development Life Cycle. The importance of this research lies in its focused examination of information security at the SDLC implementation stage [2]. This stage, often overlooked, is critical to ensuring the overall security of software systems [3].

Achieving secure software is a challenging undertaking, and research has shown that improving software development processes can effectively reduce the prevalence of vulnerabilities. However, the SSDLC process encompasses a multitude of security practices and activities aimed at achieving security objectives. The proper adoption of these activities to improve software security represents a critical concern [4]. The benefits of this research to scientific improvement are multifold. Provides a nuanced understanding of implementation-stage security practices, proposes a novel assurance model for software development vendors, and offers actionable steps and insights to mitigate information security concerns. This comprehensive approach seeks to augment the existing body of knowledge and contribute robust strategies for future developments in secure software practices [1], [3], [5], [6].

This paper specifically focuses on the aspect of information security during the implementation stage of the System Development Life Cycle (SDLC). The implementation stage of the Software Development

Received: 02-09-2023 | Accepted: 29-01-2024 | Published Online: 18-02-2024

Life Cycle (SDLC) is a critical phase in which information security plays a crucial role in ensuring the success of subsequent stages and protecting sensitive data [1]. Addressing the criticality of this stage is paramount, as it directly influences the robustness and reliability of software systems. By identifying and mitigating risks at this point, research contributes significantly to the broader field of information security and offers a template for improving practices in various development environments [7], [8].

Addressing vulnerabilities and ensuring information security during implementation can be challenging, as they often stem from user behavior rather than technical deficiencies within software [9]. Previous research has explored various factors that encompass both technical and social perspectives to achieve a secure software system [1]-[3], [5], [6], [9]-[18].

SDLC is a structured framework used by organizations to plan, design, implement, test, deploy, and maintain software systems, as presented in Figure 1. SDLC is comprised of several distinct phases, each serving a specific purpose. One of the fundamental phases of the SDLC is the implementation stage. The implementation stage is the phase in which the system is installed and deployed in its intended business environment.

This includes activities such as user training, hardware and software installation, and integration of the system into daily operational processes. During this phase, the performance of the system is closely monitored and compared to the performance objectives set during the planning phase. This stage continues until the system is fully operational and aligns with the defined user requirements [19], [20].

Figure 1. System Development Life Cycle (SDLC)

Addressing this stage is of paramount importance because the success of implementation is intricately tied to the subsequent maintenance and protection of information. In reality, many instances of information leakage and security breaches are identified during this phase. For example, the implementation stage may involve resolving issues such as the management of scattered paper records by implementing a more centralized and organized digital system to improve data management [20]. Additionally, it may involve measures to prevent and detect the presence of viruses on flash drives used for data transfer, as well as procedures to ensure data integrity and prevent data duplication [21], [22]. These proactive steps are instrumental in ensuring that the system not only meets user requirements, but also offers a more efficient and reliable means of managing information [23].

The novelty of this research lies in its systematic literature review approach, which gathers data from multiple digital repositories such as IEEE Xplore, ACM Digital Library, Scopus, and ScienceDirect. The general objective of the study is to provide a comprehensive overview of existing research on software security and identify knowledge gaps [1]. This research also aims to contribute to the field by proposing a novel assurance model for software development providers, which focuses on improving information security measures during the implementation stage [11]. By comparing its findings with existing research, this study offers valuable insights and recommendations to improve information security practices during this critical phase of SDLC [6].

Research uses a problem analysis method using the "5W1H" framework [24]. This method serves as a structured guide that features a set of essential questions to facilitate information gathering and problem resolution. These questions include: Why is the preservation of information security crucial during the implementation stage of the SDLC process? Where within the SDLC implementation stage are the specific areas that demand information security protection? When is the need for information security established and executed during the implementation stage of the SDLC process? Who plays a role in maintaining information security during the implementation stage of the SDLC process? What measures are required to protect information security at the implementation stage of the SDLC process? How can information security be effectively ensured during the implementation stage of the SDLC process? From these inquiries, the researcher has formulated the following central research question: "What are the actionable steps that can be taken to mitigate information security concerns during the implementation stage, ultimately contributing to the achievement of a successful implementation?"

2. Research Methods

A systematic review of the literature (SLR) constitutes a method for the synthesis and concise summarization of findings derived from existing research on a specific topic or research question. It embodies a systematic and transparent approach to the identification, assessment, and amalgamation of available evidence relevant to a given research inquiry or subject matter. The primary aim of an SLR is to provide a comprehensive overview of the prevailing body of knowledge concerning a particular topic while also pinpointing areas within the existing research that warrant exploration in subsequent

studies. According to Kitchenham [25], the process of performing an SLR typically includes three main phases, as delineated in Table 1.

Table 1. Phases of SLR

Phase	Stages
Planning	Identification of need
	Specifying research question(s)
	Developing review protocol
Conducting	Selection of primary studies
	Data extraction and monitoring
	Data Synthesis
Reporting	Specifying Dissemination Mechanism
	Formatting the main report
	Evaluating the report
	Documenting

The discussion is about the basic explanation, relationship, and generalization shown by the results. The description answers a research question. If there are dubious results, then show them objectively.

2.1. Planning

Planning involves several critical components. First, it involves defining the research question or topic of interest and subsequently establishing the inclusion and exclusion criteria for the studies to be included in the review. At the same time, it requires the development of a meticulous search strategy to effectively identify relevant studies [25].

Identification of Need: The initial step in planning a research study revolves around identifying the need for research. This includes the recognition of a gap within the current body of knowledge or understanding [26]. It signifies the determination that further research is indispensable to effectively address this gap.

Specifying Research Question(s): Once the need for research is discerned, the subsequent step involves the precise specification of the research question(s) that the study seeks to resolve. These questions must be sharply focused, explicitly defined, and intimately informed by the identified knowledge or understanding gap [25], [27].

Developing Review Protocol: After the research question(s) are meticulously delineated, the subsequent phase involves the formulation of a comprehensive review protocol. This protocol meticulously describes the strategic steps that will guide the research process [28]. It should encompass a detailed blueprint for data collection and analysis, coupled with a well-structured timeline that governs the completion of various research stages.

Furthermore, the review protocol should include a robust strategy to spread the research results, potentially through avenues such as publication in scientific journals or presentation at conferences.

2.2. Data Source

This research employs an automated search technique to systematically gather data from multiple digital repositories. The use of automated search techniques is a common practice among researchers when collecting data from online sources, as it offers greater efficiency and completeness when searching for relevant literature [29].

The construction of the search string used in the automated search process is of paramount importance. It must be meticulously designed to ensure the identification of the most relevant literature. Furthermore, the selection of repositories for search is contingent on their relevance to the research topic and the quality of the information they house. It is essential to rigorously assess the information sources to determine their reliability and relevance to the ongoing research effort [30].

The following digital sources have been chosen for this study: IEEE Xplore (IEEE) = 38, ACM Digital Library (ACM) = 44 papers, Scopus (Scopus) = 4, ScienceDirect (SD) = 0. These repositories have been selected based on their alignment with the research topic and their proven quality in containing pertinent information.

2.3. Search String

Search strings were generated using keywords derived from research questions and existing literature [10]. This study used Boolean “OR” and “AND” operators to concatenate the keywords into search strings. The following string was used to scan the digital repositories: ("implementation" AND ("phase" OR "stage")) AND "information security" AND ("SDLC" OR "System Development Life Cycle"). Our data inclusion criteria align with the parameters established by previous researchers. To determine data exclusion, we adhered to guidelines based on established criteria from previous research. These conditions are shown in Table 2.

Table 2. Criteria for Inclusion and Exclusion Data

Inclusion Criteria	Exclusion Criteria
Articles written in English.	Papers not written in English.
Papers published between 2012 and 2023.	Duplicate papers were excluded.
Articles related to the domain of software engineering.	Papers that lack detailed descriptions of software security risks in software development.
The articles must provide at least one risk or practice relevant to the software development process, specifically the coding phase.	Papers that do not address software risks in software development and are not relevant to the research questions.
The articles were peer-reviewed in conferences and journals.	Publications that are not peer reviewed and do not constitute complete books, abstracts, editorials, or letters.

These stringent inclusion and exclusion criteria were diligently applied to ensure data selection that aligns closely with research objectives and quality standards.

2.4. Conducting

The conducting phase, as described by Kitchenham et al. [25], encompasses active search and identification of pertinent research studies, critical evaluation of their quality and relevance, and systematic extraction of data that are of significance for research. Furthermore, Khan et al. [26] provide information on identifying relevant studies in software engineering, which is essential for a comprehensive research process.

The research articles identified during the selection of the primary study underwent a meticulous refinement process using the Tollgate approach presented by Mousa et al. [28]. Table 3 outlines this approach, which is structured into three pivotal phases.

Phase 1 (Ph-1): In this initial phase, the quest involves finding relevant articles through the application of well-defined search terms.

Phase 2 (Ph-2): The subsequent phase involves a meticulous evaluation of articles based on criteria such as title, abstract, and keywords, allowing the inclusion of articles that meet the predefined criteria while excluding those that do not.

Phase 3 (Ph-3): In this conclusive phase, the inclusion and exclusion process is further refined, this time based on a comprehensive examination of the full-text content. It culminates in the definitive selection of primary studies that are to be included in the systematic review of the literature (SLR).

Table 3. Selection of articles using the tollgate approach

Database	Ph-1	Ph-2	Ph-3
IEEE	i=38; e=0	i=34; e=4	i=31; e=3
ACM	i=44; e=0	i=38; e=6	i=33; e=5
Scopus	i=4; e=0	i=4; e=0	i=3; e=1
SD	i=0; e=0	i=0; e=0	i=0; e=0
Total	i=86; e=0	i=76; e=10	i=67; e=9

Legend: i = included papers; e = excluded papers.

2.5. Reporting

The reporting phase, as described by Kitchenham et al. [25], constitutes the culmination of the systematic literature review process. During this phase, the data gleaned from the selected studies are synthesized and meticulously presented clearly and concisely. This presentation includes a detailed exploration of the review findings, including a discussion of their implications. Furthermore, this phase may also provide recommendations for future research endeavors based on the insights derived from the review.

In essence, the overarching objective of a systematic literature review (SLR) is to provide a comprehensive and contemporaneous overview of the existing body of research on a specific topic. Simultaneously, it aims to pinpoint the gaps in the current knowledge landscape, thus paving the way for prospective studies that can further contribute to the advancement of understanding within the field [27].

2.6. Comparison Studies

The methodology employed in this study is grounded in an extensive review of the literature. The research was carried out through a structured sequence of phases, described as follows.

Software security is of great importance. Often, security considerations are postponed until after the software has been fully developed, with minimal attention given to security in the early stages of the software development life cycle (SDLC). Regrettably, there is currently no established methodology for quantifying the security of SDLC artifacts when security is integrated from the very beginning of the software development process. Assessment of the security levels for SDLC artifacts in each stage of software development requires a quantifiable approach. To mitigate vulnerabilities and improve security within software applications, the allocation of resources is imperative. Quantification plays a pivotal role in aiding software developers in this endeavor.

In a recent series of articles [31]-[37], a methodology is presented that leverages vulnerability events to calculate a vulnerability index and combines this with an assessment of the potential damage attributable to these vulnerabilities to determine a comprehensive security index. This approach provides valuable information on the security landscape, offering a foundation on which software developers can make informed decisions to improve security throughout the development process.

The comparative study presented in the articles [8], [22], [38]- [58] serves as a valuable guide for software developers in selecting specific methodologies for building secure software applications. In this study, we conduct a thorough comparison and contrast of various development processes, focusing on key characteristics essential to an effective secure software development process.

In addition, this paper conducts an in-depth analysis of the desirable attributes associated with security specification languages. Furthermore, to obtain complete security requirements, identification of activities within the security requirements engineering process is imperative. On the basis of this foundation, the research compares the engineering processes of various security requirements methodologies. The analysis reveals that certain desired properties, critical to many secure software requirements engineering methods, are absent from some of the methodologies under examination.

The objective of the research articles [59], [60], [61], [62], [26], [63], [64], [65], [66], [67], [68] was to comprehensively assess the landscape of secure software development through the implementation of a systematic mapping study (SMS). This SMS included the identification of pertinent literature based on rigorous inclusion and exclusion criteria, the extraction

of pertinent data from these sources, and the subsequent classification of these articles using various criteria. These criteria included considerations such as quality assessment, software security methodologies, software development life cycle (SDLC) phases, publication venues, and SWOT analysis (Strengths, Weaknesses, Opportunities, and Threats) analysis.

This comprehensive study sheds light on a compelling landscape – one that calls for intensive research in the field of Secure Software Development (SSD) [59], [60], [61], [62], [26], [63], [64], [65]. In particular, research reveals a dearth of empirically validated solutions, indicating a substantial gap between theoretical security measures and their application and validation in real-world scenarios [66], [67], [68].

This gap underscores the dynamic nature of secure software engineering, where advances in theory and practice must continuously adapt to the evolving landscape of threats and technological advancements [65]. Consequently, the study findings highlight the urgent need for continuous refinement of security measures and strategies, advocating a more empirical approach to validate and refine these methods in practical contexts [69].

In essence, the call for further research, as illuminated by this study, transcends mere academic pursuit. It represents a critical imperative for both industry and academia to prioritize the empirical validation of security methodologies [60], [61] [62] [26] [63] [64] [65] [66] [67]. By embracing this approach, we can ensure that theoretical models and strategies are not only conceptually sound but also demonstrably effective in practice, ultimately paving the way for the development of robust and resilient secure software systems [65].

Software vulnerabilities can emanate from diverse sources, encompassing technical flaws within the software's design or implementation, as well as insecure practices exhibited by users. User-side vulnerabilities arise when users fail to adhere to established security protocols and practices. Addressing these vulnerabilities can be challenging, as they often stem from user behavior rather than inherent technical deficiencies within the software.

In articles [1]-[3], [5], [6], [9]-[18], a comprehensive exploration is carried out to elucidate the multifaceted factors that encompass both technical and social perspectives. These factors are crucial considerations in achieving a secure software system, necessitating a delicate equilibrium between technical fortifications and user-centered security measures.

Prioritizing security within the design phase of the software development life cycle (SDLC) is of paramount importance for software development organizations. To facilitate this crucial emphasis on security, a Secure Software Design Maturity Model (SSDMM) has been developed. This model serves as an invaluable tool for evaluating and enhancing an organization's security practices throughout the design phase of the SDLC. SSDMM functions as a comprehensive framework, allowing organizations to assess their maturity levels with respect to secure design practices, as outlined in articles [23], [69], [70], [71]. By employing this model, software development organizations can systematically advance their security posture and ensure the integration of robust security measures during the design phase of their software development processes.

Cloud computing has emerged as a widely preferred platform for fostering innovative applications, but it is not immune to inherent risks and vulnerabilities throughout its life cycle. In response to these challenges, it becomes imperative to establish a comprehensive framework for self-governing cloud security, one that spans all phases of the SDLC. This framework, aptly named the Cloud Secure Software Development Life Cycle (Cloud SSDLC), seamlessly incorporates essential cloud security domains and their associated risks into every facet of the SDLC. Within the planning phase of Cloud SSDLC, a critical undertaking is the identification of security requirements and the attendant risks specific to the cloud application.

Subsequently, a comprehensive plan is devised to address these identified risks. This may require a thorough risk assessment and the identification of potential vulnerabilities. Additionally, it involves the development of the necessary security controls and procedural measures to effectively mitigate these identified risks. In its entirety, the cloud SSDLC framework serves as an instrumental means to ensure that cloud applications are designed and maintained with the utmost security in mind, thus diminishing susceptibility to vulnerabilities and potential security threats, as elaborated in articles[21], [72]-[75].

In several studies, [8], [64], [76] R.A. Khan et al. underscored the criticality of security as an integral facet of software quality. In recent years, the frequency and impact of security attacks have increased significantly. Consequently, there is a pressing need for new paradigms of software development to create inherently secure software [33]. Regrettably, numerous organizations still relegate security to an afterthought, perpetuating persistent security vulnerabilities. Integration of security measures into the Software Development Lifecycle (SDLC) has become an urgent imperative, with a plethora of methodologies, strategies, and models proposed [35].

However, only a select few of these approaches are supported by credible evidence to promote the development of genuinely secure software applications [43]. Effectively integrating security protocols into the SDLC continues to pose a formidable challenge.

In a separate investigation [77], conducted by Mazni Mohamed Jakeri and Mohd Fadzil Hassan, the pivotal role of security in protecting data from unauthorized access was underscored. In response to security concerns, multiple security frameworks have been introduced for the Secure Software Development Life Cycle (SSDLC). Secure SDLC is achieved by integrating security-related activities into each phase of widely employed development methodologies, such as the Waterfall or Agile models. However, these frameworks often face under-use due to factors such as rigidity, complexity, and resource intensity [4], [9], [36], [65], [78]. The consensus remains that integrating security, particularly during the requirements and design phases, represents the most effective and cost-effective approach to developing secure web applications. The disparities resulting from the identification of key variables in previous research articles are delineated in Table 4.

Table 4. Identification Key Variables

Literature	Variables																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
[1]	✓					✓					✓					✓			
[9]	✓					✓					✓					✓			
[10]	✓				✓						✓					✓			
[11]	✓	✓	✓	✓	✓	✓												✓	
[3]							✓				✓	✓		✓					
[14]	✓		✓		✓											✓			
[2]	✓	✓	✓	✓	✓	✓							✓			✓			
[31]	✓	✓	✓	✓	✓	✓													
[33]	✓	✓	✓	✓	✓	✓										✓		✓	
[34]	✓	✓	✓	✓	✓	✓										✓		✓	
[35]	✓	✓											✓						
[36]		✓											✓			✓			
[38]			✓									✓				✓		✓	
[39]	✓	✓											✓						
[44]							✓	✓	✓	✓		✓							
[45]			✓								✓							✓	
[47]			✓	✓							✓								
[34]	✓				✓							✓							
[50]									✓	✓	✓	✓							
[22]				✓					✓	✓	✓	✓							
[8]	✓				✓						✓	✓	✓	✓		✓			
[55]									✓	✓	✓	✓							
[56]									✓	✓	✓	✓							
[57]	✓	✓	✓	✓	✓							✓		✓					
[59]									✓	✓	✓								
[60]	✓	✓	✓	✓	✓	✓	✓					✓			✓		✓	✓	
[65]							✓	✓	✓			✓		✓					
[66]	✓				✓		✓				✓	✓		✓					✓
[68]	✓										✓		✓						
[69]		✓	✓	✓	✓	✓											✓	✓	

Literature	Variables																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
[78]																			✓
[23]			✓		✓	✓	✓	✓				✓		✓		✓	✓	✓	✓
[72]	✓		✓		✓	✓						✓	✓				✓	✓	✓
[74]	✓		✓		✓	✓						✓	✓				✓	✓	✓
[4]	✓		✓		✓	✓						✓					✓	✓	
[19]	✓		✓		✓	✓	✓	✓				✓		✓			✓	✓	
[20]	✓		✓		✓	✓						✓					✓	✓	
[77]												✓			✓				✓

This research introduces a novel perspective by specifically honing in on the implementation stage within the Software Development Life Cycle (SDLC), distinguishing itself from previous related studies that encompassed all phases of the SDLC [22], [31], [47], [57]-[71]. We provide a comprehensive delineation of the intricacies involved in executing the SDLC. It constitutes an invaluable resource for those seeking to enhance the success of their implementation efforts. The correlation of the literature, as well as the grouping of identified key variables, is illustrated in Table 5, with the corresponding references provided in the reference list.

Table 5. Code of Variables

Variable	Description
1	Security requirements
2	Security analysis
3	Security design, Security Architecture, Security configuration
4	Security implementation
5	Security testing
6	Security monitoring
7	Security training
8	Security tools
9	Security policies
10	Security procedures
11	Security control
12	Security awareness
13	Security assurance
14	Security culture
15	Security budget, Security cost, Security Investment
16	Security risk analysis
17	Security incident response
18	Security mitigation, Security mechanism
19	Security measures, Security metrics

3. Results and Discussions

This chapter systematically addresses the questions that have arisen within the scope of this study, providing comprehensive responses.

3.1. Why is information security protection vital during the implementation stage in the SDLC process?

Ensuring the security of information during the implementation stage of the Software Development Life Cycle (SDLC) is of paramount importance. This phase marks the actual construction and deployment of the software, making it a pivotal point for protecting against vulnerabilities and weaknesses [38]-[61]. Several compelling reasons underscore the importance of information security protection at this stage.

Protection of sensitive data: In cases where software handles sensitive data, such as financial or personal information, it becomes imperative to secure these data, protecting them from unauthorized access or manipulation [38].

Prevention of vulnerabilities and cyber-attacks: Software that lacks robust security measures can become susceptible to various forms of cyber threats, including malware, ransomware, and other malicious attacks [38]-[61]. These threats pose severe risks, potentially compromising the software's integrity and resulting in repercussions such as data loss or theft.

Compliance obligations: Many organizations are bound by legal and regulatory mandates on information security, exemplified by regulations such as the General Data Protection Regulation (GDPR) and the Payment Card Industry Data Security Standard (PCI DSS) [38]-[61]. Non-compliance with these regulations can result in fines, legal penalties, and substantial reputation damage.

Taking proactive measures to protect information security during the implementation stage allows organizations to fortify their software security posture, adhere to compliance standards, and protect sensitive data effectively [38].

3.2. Where should information security be applied within the implementation stages of the SDLC process?

Within the software development life cycle (SDLC) process, the implementation stage encompasses various critical aspects that require robust information security measures [59] - [75]. These include the following.

Codebase: The codebase serves as the central repository that houses all the source code relevant to a software project [62], [72]. Preserving the integrity of the codebase is essential to protect against unauthorized access or tampering, as such breaches can introduce vulnerabilities in the software [73]-[75].

Build and Test Environments: Build and test environments are integral to compiling, building, and rigorously testing software [59], [60]. It is imperative to fortify these environments against unauthorized access or tampering, as any such compromise could result in the injection of vulnerabilities into software [72]-[76].

Deployment and Production Environments: Deployment and production environments signify the arenas where the software is ultimately deployed and utilized by end-users. Ensuring the security of these environments is paramount to thwart unauthorized access or tampering, as any breach could compromise both the integrity and security of the software [61], [62], [72]-[76].

Communication and Collaboration Tools: Many software development teams rely on communication and collaboration tools, including chat and project management software, to facilitate project collaboration [72]. Protection of these tools against unauthorized access or manipulation is crucial to maintaining the security of the entire software development process [59]- [62], [73]- [76].

By diligently protecting these critical components within the SDLC process implementation stage, organizations can significantly improve the security and compliance of their software while also ensuring the protection of sensitive data [59]- [71], [72]-[74].

3.3. When is information security essential during the SDLC process implementation stage?

Information security should be an integral consideration and practice throughout the software development life cycle (SDLC) process, with particular emphasis on the implementation stage. This phase marks the active construction and deployment of the software, making it a pivotal juncture to guarantee the software's security while identifying and mitigating vulnerabilities or weaknesses. Several key junctures within the implementation stage necessitate a steadfast commitment to information security [59]-[71]:

Commencement of the implementation stage: At the start of the implementation stage, it is imperative to define the security requirements of the software. Ensuring that these requirements are fully integrated into design and development processes is critical [58], [73].

During the Coding Process: Throughout the development phase, strict adherence to secure coding practices is imperative. Detecting and addressing any vulnerabilities or weaknesses within the code is essential during this stage [27], [61].

Throughout Testing: As the software undergoes rigorous testing, the inclusion of security testing is paramount. This step helps to systematically uncover and rectify any vulnerabilities or weaknesses that may exist [4], [19].

Pre-Deployment: Before deployment, meticulous tests should follow to confirm that the software has been rigorously evaluated, with all vulnerabilities and weaknesses diligently addressed [60], [69].

By steadfastly implementing information security measures at these pivotal points within the SDLC process implementation stage, organizations can substantially fortify the security and compliance of their software, while simultaneously protecting sensitive data [58].

3.4. Who participates in protecting information security during the implementation stage of the SDLC process?

The protection of information security within the implementation stage of the software development life cycle (SDLC) process typically requires the collaboration of a team consisting of professionals with distinct roles and responsibilities. Several key individuals who may play pivotal roles in maintaining information security during this stage include [59] - [62]:

Software Developers: Software developers assume the responsibility of coding and building software. Their proficiency should extend to secure coding practices and have a deep understanding of the importance of integrating security into the software development process [78].

Quality Assurance (QA) Testers: QA testers are charged with the task of meticulously evaluating the software to ensure its quality and adherence to the specified requirements. Proficiency in security testing techniques is imperative, along with greater awareness of the importance of identifying and rectifying vulnerabilities or weaknesses in software [23].

Information Security Professionals: Information security experts, including security analysts or security engineers, may contribute by reviewing the software for vulnerabilities and weaknesses. They are instrumental in the development and implementation of security measures designed to protect software [70].

Project Managers: Project managers have the responsibility of overseeing the entire software development process, ensuring its timely completion and adherence to budget constraints. They should also recognize the paramount importance of integrating security measures into the process and take measures to ensure that all team members are well versed in their security responsibilities [71].

Through collaborative efforts, these professionals can effectively fortify the security and compliance of the software, ultimately protecting sensitive data throughout the SDLC process implementation stage [70].

3.5. What is required to protect information security during the implementation stage of the SDLC process?

To maintain information security during the implementation stage of the software development life cycle (SDLC) process, several imperative measures must be diligently embraced [31]-[37]:

Secure coding practices: Ensuring that software developers are well versed in secure coding practices is fundamental. They must adhere to industry best practices when developing software, thus avoiding the introduction of vulnerabilities or weaknesses into the software architecture [35].

Security testing: Integrating comprehensive security testing is paramount within the testing process. Security testing encompasses techniques such as vulnerability detection, penetration testing, and code review. These practices are essential for the identification and subsequent mitigation of vulnerabilities or weaknesses within the software [36].

Secure development environment: The development environment itself must be fortified against unauthorized access or manipulation. This may involve the implementation of strict access controls, strengthened network security protocols, and robust data encryption mechanisms [33].

Secure deployment and production environments: Similarly, to the development environment, the deployment and production environments must also be secured to prevent unauthorized access or tampering. This requires the enforcement of robust access controls,

comprehensive network security measures, and vigilant data encryption practices [37].

By implementing these measures firmly, organizations can actively ensure that their software remains securely anchored, in compliance with relevant standards, and secure in protecting sensitive data throughout the SDLC process implementation stage [36].

3.6. How can information security be safeguarded during the implementation stage of the SDLC process?

Preserving information security during the implementation stage of the software development life cycle (SDLC) process requires the adhesion to a series of strategic steps [1], [8], [10]-[18], [42], [48], [50],:

Establish security requirements: Right at the commencement of the implementation stage, a pivotal step is to define the software's security requirements. Integration of these requirements into design and development processes is essential to establish a strong foundation for security [16].

Adopt secure coding practices: Software developers must have a thorough understanding of secure coding practices. Their adhesion to these practices during software coding can effectively prevent the introduction of vulnerabilities or weaknesses [12].

Integrate security testing: Comprehensive security testing must be seamlessly integrated into the testing phase. Techniques such as vulnerability scanning, penetration testing, and code review are crucial to identifying and subsequently addressing vulnerabilities or weaknesses within software [8], [42].

Leverage a secure development environment: Fortifying the development environment against unauthorized access or tampering is critical. Implementing robust measures such as access controls, network security fortifications, and data encryption is crucial to maintaining the integrity of the environment [6].

Ensure a secure deployment environment: The deployment and production environments must mirror a similar level of security to thwart unauthorized access or tampering. Implementing stringent access controls, enhanced network security, and vigilant data encryption practices is imperative [48].

By diligently following these strategic steps, organizations can assertively safeguard their software's security and compliance. This robust approach ensures that sensitive data is kept consistently protected throughout the SDLC process implementation stage [10].

3.7. What actionable steps can be taken to mitigate information security concerns during the implementation stage, ultimately contributing to the achievement of a successful implementation?

The culmination of responses to the aforementioned inquiries has culminated in a comprehensive summary that addresses the central research question posited in this paper. When comparing our research with the body of work presented in the related research, we have identified nuanced insights and perspectives that further enrich our understanding of this domain. Although previous studies have provided valuable information, this research sheds additional light on the specific challenges, practices, and actionable steps related to information security during the implementation stage. It also describes the necessary steps that must be meticulously carried out during the implementation stage [35].

This nuanced perspective contributes to the growing body of knowledge surrounding software security and implementation practices[21], [65], and also underscores the importance of addressing information security comprehensively throughout SDLC [77], particularly during the implementation stage, as highlighted by the findings in Table 6. It is important to note that the insights presented in Table 6 are the result of a meticulous synthesis of existing literature. Although not directly referenced, they represent a valuable compilation of security measures and practices from various sources.

Table 6. 12 Steps Implementation Stage

Steps	Literature
Coordination meeting	[1], [2], [4], [8], [9]-[11], [19], [20], [33]-[36], [39], [49], [57], [68], [72], [74],
Implementation survey	[2], [4], [5], [19], [20], [22], [23], [31], [39], [44], [55], [56], [59], [65], [72], [74],
Installation	[1], [2], [3], [8], [9]-[11], [14], [19], [22], [23], [33]-[36], [39], [44], [47], [49], [55], [57], [60], [79]
Kick-off meeting	[1]-[3], [4], [8], [19], [20], [33]-[36], [39], [49], [57], [66], [68], [72], [74]
General explanation	[1]-[3], [4], [8], [14], [19], [22], [23], [31], [33]-[35], [39], [44], [47], [49], [55], [57], [60], [79]
Inter-department training	[2], [19], [20], [22], [31], [39], [44], [55], [56], [59], [65], [72], [74], [79]
Operator training	[4], [19], [20], [23], [65], [66], [72], [74]
Enter master & and static data	[1], [14], [36], [38], [44], [66], [72], [74], [78]
System live	[1], [2], [3], [4], [8], [14], [19], [20], [22], [31], [33]-[35], [38], [39], [44], [45], [47], [49], [56], [57], [59], [60], [66], [68], [69], [72], [74], [78], [79]
Implementation assistance	[8], [19], [23], [28], [57], [65], [66], [69], [78]
System audit	[58], [59], [70], [71], [75], [78],
Close off meeting	[1]-[3], [4], [8], [19], [20], [33]-[35], [39], [49], [57], [66], [68], [72], [74]

Table 6 presents the steps to mitigate information security issues during the SDLC implementation stage.

Coordination meeting: Arrange and approve the implementation schedule between the provider and the customer. In this meeting, both parties should introduce each other's team and Person in Charge (PIC) [72].

Implementation survey: In this step, the provider's team will conduct a detailed survey of all existing procedures, policies, and the availability of master & and static data [56].

Installation: The Provider will install the software on the Customer's server. The installation of the operating system and database is the responsibility of the customer [23].

Kick-off meeting: This meeting is held between the Provider's Management and Implementation team and the Customer's Management team to indicate the start of the Implementation process [10].

General explanation: In this session, the Provider will provide a general explanation of the Software and its background. This session aims to build good cooperation between the provider and the customer [10].

Inter-department training: The Provider explains the flow of the system and the corresponding policies and procedures of each department. During this session, the customer will gain a complete understanding of the new policies and procedures that will be implemented along with the system [44].

Operator training: The employees are trained according to their level of operational function. This training ensures that every employee understands how to enter transactions using the system [21].

Enter master & static data: In this step, the customer will input Master & Staand static into the system. Before going live, all necessary preparations and data must be entered into the system. The complexity of this step varies depending on the number of transactions and the availability of data [59].

System live: When all the above preparations have been completed and all required data have been input into the system, the Customer is ready for System Live. This means that the customer will enter all their daily transactions using the system. When all transactions are up to date, all reports can be automatically generated from the system [15].

Implementation assistance: During this period, the implementation team will continue to assist the customer to ensure that the customer can operate the system smoothly [74].

System audit: Near the end of the implementation process, the Provider's team will perform a system audit of each user to make sure that every user fully understands the proper use of the system [4].

Close-off meeting: This last meeting marks the end of the implementation process. This means that the customer will now enter the maintenance period, which will be handled by the maintenance team [79].

4. Conclusions

In conclusion, this research has successfully addressed critical aspects of information security within the Software Development Life Cycle (SDLC), with a specific focus on the implementation stage. Through the structured 5W1H framework, the study has provided comprehensive information on why information security is vital at this stage, where and when it should be implemented, who plays crucial roles, what measures are necessary, and how effective security can be ensured.

Research emphasizes the paramount importance of protecting information security during implementation, as it directly influences the success of subsequent stages and the protection of sensitive data. Various actionable steps and strategies have been outlined to mitigate security concerns during this phase, which ultimately contributes to the achievement of a successful implementation.

This study has provided actionable steps and insights on mitigating information security concerns during implementation, contributing to the achievement of a successful implementation. These findings not only serve as valuable guidance for organizations, but also offer opportunities for further comprehensive research in this critical domain of software development. The recommendations arising from this study advocate for the continuous adaptation and assimilation of emerging technologies and methodologies into SDLC as a means of strengthening security postures.

This requires empirical research to rigorously evaluate the effectiveness of integrated security measures within diverse development environments and across industry sectors. Furthermore, the study recommends the development of robust and adaptable security frameworks capable of seamless integration into existing and future SDLC models. This fosters an ecosystem where security co-evolves with technological advancements, ensuring ongoing resilience against evolving threats.

References

[1] A. Ramirez, A. Aiello, and S. J. Lincke, "A Survey and Comparison of Secure Software Development Standards," in *2020 13th CMI Conference on Cybersecurity and Privacy (CMI) - Digital Transformation - Potentials and Challenges(51275)*, 2020, pp. 1–6, doi: 10.1109/CMI51275.2020.9322704.

[2] J. C. S. Núñez, A. C. Lindo, and P. G. Rodríguez, "A Preventive Secure Software Development Model for a Software Factory: A Case Study," *IEEE Access*, vol. 8, pp. 77653–77665, 2020, doi: 10.1109/ACCESS.2020.2989113.

[3] T. Thomas, M. Tabassum, B. Chu, and H. Richter Lipford, *Security During Application Development: an Application Security Expert Perspective*. 2018.

[4] K. Meridji, K. T. Al-Sarayreh, A. Abran, and S. Trudel, "System security requirements: A framework for early

identification, specification and measurement of related software requirements," *Comput. Stand. Interfaces*, vol. 66, p. 103346, 2019, doi: https://doi.org/10.1016/j.csi.2019.04.005.

[5] M. Baldassarre, V. Barletta, D. Caivano, and A. Piccinno, *Integrating Security and Privacy in HCD-Scrum*. 2021.

[6] A. B. Ajmal, M. A. Shah, C. Maple, M. N. Asghar, and S. U. Islam, "Offensive Security: Towards Proactive Threat Hunting via Adversary Emulation," *IEEE Access*, vol. 9, pp. 126023–126033, 2021, doi: 10.1109/ACCESS.2021.3104260.

[7] T. Lopez, H. Sharp, T. Tun, A. Bandara, M. Levine, and B. Nuseibeh, "Security Responses in Software Development," *ACM Trans. Softw. Eng. Methodol.*, Sep. 2022, doi: 10.1145/3563211.

[8] R. A. Khan, S. U. Khan, M. Alzahrani, and M. Ilyas, "Security Assurance Model of Software Development for Global Software Development Vendors," *IEEE Access*, vol. 10, pp. 58458–58487, 2022, doi: 10.1109/ACCESS.2022.3178301.

[9] L. N. Q. Do, J. R. Wright, and K. Ali, "Why Do Software Developers Use Static Analysis Tools? A User-Centered Study of Developer Needs and Motivations," *IEEE Trans. Softw. Eng.*, vol. 48, no. 3, pp. 835–847, 2022, doi: 10.1109/TSE.2020.3004525.

[10] A. Garg, R. K. Kaliyar, and A. Goswami, "PDRSD-A systematic review on plan-driven SDLC models for software development," in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2022, vol. 1, pp. 739–744, doi: 10.1109/ICACCS54159.2022.9785261.

[11] D. Stewart, "Security versus Compliance: An Empirical Study of the Impact of Industry Standards Compliance on Application Security," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 32, pp. 1–31, Apr. 2022, doi: 10.1142/S0218194022500152.

[12] B. Aljedaani, A. Ahmad, M. Zahedi, and M. A. Babar, "Security Awareness of End-Users of Mobile Health Applications: An Empirical Study," *CoRR*, vol. abs/2008.1, 2020, [Online]. Available: https://arxiv.org/abs/2008.13009.

[13] M. Unal and O. Bolukbas, *The Acquirements of Digitalization with RPA (Robotic Process Automation) Technology in the Vakif Participation Bank*. 2021.

[14] M. Kang and A. Hovav, "Benchmarking Methodology for Information Security Policy (BMISP): Artifact Development and Evaluation," *Inf. Syst. Front.*, May 2018, doi: 1.1007/s1079.

[15] S. G.C, T. Sake, and . A., "A Systematic Review and Catalog of Security Metric during the Secure Software Development Life Cycle," *Recent Adv. Electr. Electron. Eng. (Formerly Recent Patents Electr. Electron. Eng.*, vol. 13, Dec. 2020, doi: 10.2174/2352096513999201201121823.

[16] Y. Perdana and D. I. Sensuse, "Knowledge Sharing System Development: A Systematic Literature Review," in *2021 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2021, pp. 1–7, doi: 10.1109/ICACSIS53237.2021.9631327.

[17] R. E. Fairley, "Traditional Process Models for System Development," in *Systems Engineering of Software-Enabled Systems*, IEEE, 2019, pp. 99–119.

[18] S. Sheikhi and P. Kostakos, "Cyber threat hunting using unsupervised federated learning and adversary emulation," in *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2023, pp. 315–320, doi: 10.1109/CSR57506.2023.10224990.

[19] R. Fujdiak *et al.*, "Managing the Secure Software Development," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2019, pp. 1–4, doi: 10.1109/NTMS.2019.8763845.

[20] Michigan Technological University, "System development lifecycle (SDLC): Information Technology: Michigan Tech." [Online]. Available: https://www.mtu.edu/it/security/policies-procedures-guidelines/information-security-program/system-development-lifecycle/.

[21] A. D. Bhagat, S. Basia, K. Sharma, and P. Vats, "A Survey of Cloud Architectures: Confidentiality, Contemporary State, and Future Challenges," in *2022 3rd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, 2022, pp. 1–8, doi: 10.1109/ICICT55121.2022.10064580.

[22] T. Eom, J. Hong, S. An, J. Park, and D. Kim, "A Systematic Approach to Threat Modeling and Security Analysis for Software Defined Networking," *IEEE Access*, vol. PP, p. 1, Sep. 2019, doi: 10.1109/ACCESS.2019.2940039.

[23] H. Al-Matouq, S. Mahmood, M. Alshayeb, and M. Niazi, "A Maturity Model for Secure Software Design: A Multivocal Study," *IEEE Access*, vol. 8, pp. 215758–215776, Jan. 2020, doi: 10.1109/ACCESS.2020.3040220.

[24] C. Quinlan, W. Zikmund, B. Babbin, J. Carr, and M. Griffin, *Business Research Methods*. 2015.

[25] B. Kitchenham, L. Madeyski, and D. Budgen, "SEGRESS: Software Engineering Guidelines for REporting Secondary Studies," *IEEE Trans. Softw. Eng.*, vol. PP, p. 1, Jan. 2022, doi: 10.1109/TSE.2022.3174092.

[26] D.-R. Khan, S. U. Khan, and M. Ilyas, *Exploring Security Procedures in Secure Software Engineering: A Systematic Mapping Study*. 2022.

[27] P. Ralph, "Toward Methodological Guidelines for Process Theories and Taxonomies in Software Engineering," *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 712–735, 2019, doi: 10.1109/TSE.2018.2796554.

[28] A. Mousa, M. Karabatak, and T. Mustafa, *Database Security Threats and Challenges*. 2020.

[29] A. A. R. Farea, C. Wang, E. Farea, and A. B. Alawi, "Cross-Site Scripting (XSS) and SQL Injection Attacks Multi-classification Using Bidirectional LSTM Recurrent Neural Network," in *2021 IEEE International Conference on Progress in Informatics and Computing (PIC)*, 2021, pp. 358–363, doi: 10.1109/PIC53636.2021.9687064.

[30] A. Shrivastava, S. Choudhary, and A. Kumar, "XSS vulnerability assessment and prevention in web application," in *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, 2016, pp. 850–853, doi: 10.1109/NGCT.2016.7877529.

[31] K. Asamoah *et al.*, "Zero-Chain: A Blockchain-Based Identity for Digital City Operating System," *IEEE Internet Things J.*, vol. PP, p. 1, Apr. 2020, doi: 10.1109/JIOT.2020.2986367.

[32] A. Agrawal *et al.*, "Software Security Estimation Using the Hybrid Fuzzy ANP-TOPSIS Approach: Design Tactics Perspective," *Symmetry (Basel).*, vol. 12, pp. 1–21, Apr. 2020, doi: 10.3390/sym12040598.

[33] R. R. Althar, D. Samanta, M. Kaur, A. A. Alnuaim, N. Aljaffan, and M. Aman Ullah, "Software Systems Security Vulnerabilities Management by Exploring the Capabilities of Language Models Using NLP.," *Comput. Intell. Neurosci.*, vol. 2021, p. 8522839, 2021, doi: 10.1155/2021/8522839.

[34] M. Ganesh, A. Xavier, B. David, M. Sagayam, and A. Elngar, *Model Transformation and Code Generation Using a Secure Business Process Model*. 2022.

[35] S. Solms and L. Futcher, "Adaption of a Secure Software Development Methodology for Secure Engineering Design," *IEEE Access*, vol. PP, p. 1, Jul. 2020, doi: 10.1109/ACCESS.2020.3007355.

[36] M. Siavvas, D. Tsoukalas, M. Jankovic, D. Kehagias, and D. Tzovaras, "Technical debt as an indicator of software security risk: a machine learning approach for software development enterprises," *Enterp. Inf. Syst.*, Sep. 2020, doi: 10.1080/17517575.2020.1824017.

[37] C.-M. Mathas, C. Vassilakis, N. Kolokotronis, C. C. Zarakovitis, and M.-A. Kourtis, "On the Design of IoT Security: Analysis of Software Vulnerabilities for Smart Grids," *Energies*, vol. 14, no. 10. 2021, doi: 10.3390/en14102818.

[38] R. R. Althar, D. Samanta, M. Kaur, D. Singh, and H.-N. Lee, "Automated Risk Management Based Software Security Vulnerabilities Management," *IEEE Access*, vol. 10, pp. 90597–90608, 2022, doi: 10.1109/ACCESS.2022.3185069.

[39] M. Jouini, L. Ben Arfa Rabai, and R. Khédri, "A quantitative assessment of security risks based on a multifaceted classification approach," *Int. J. Inf. Secur.*, vol. 20, Aug. 2021, doi: 10.1007/s10207-020-00515-6.

[40] Z. Sun, K. D. Strang, and F. Pambel, "Privacy and security in the big data paradigm," *J. Comput. Inf. Syst.*, vol. 60, pp. 1–10, Feb. 2018, doi: 10.1080/08874417.2017.1418631.

[41] A. Khan, F. Khan, J. Khan, J. Khan, and Y. Lee, *Identification and Prioritization of Critical Cyber Security Challenges and*

Practices for Software Vendor Organizations in Software Development: An AHP-Based Systematic Approach. 2022.

[42] A. Khan *et al.*, "Analyzing and Evaluating Critical Challenges and Practices for Software Vendor Organizations to Secure Big Data on Cloud Computing: An AHP-Based Systematic Approach," *IEEE Access*, vol. PP, p. 1, Jul. 2021, doi: 10.1109/ACCESS.2021.3100287.

[43] T. Lopez, H. Sharp, A. Bandara, T. Tun, M. Levine, and B. Nuseibeh, "Security Responses in Software Development," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 3, Apr. 2023, doi: 10.1145/3563211.

[44] K. Rindell, J. Ruohonen, and S. Hyrynsalmi, *Surveying Secure Software Development Practices in Finland*. 2018.

[45] H. Nina, J. A. Pow-Sang, and M. Villavicencio, "Systematic Mapping of the Literature on Secure Software Development," *IEEE Access*, vol. 9, pp. 36852–36867, 2021, doi: 10.1109/ACCESS.2021.3062388.

[46] W. Wang, F. Dumont, N. Niu, and G. Horton, "Detecting Software Security Vulnerabilities Via Requirements Dependency Analysis," *IEEE Trans. Softw. Eng.*, vol. 48, no. 5, pp. 1665–1675, 2022, doi: 10.1109/TSE.2020.3030745.

[47] M. Alenezi, A. Agrawal, R. Kumar, and R. A. Khan, "Evaluating Performance of Web Application Security Through a Fuzzy Based Hybrid Multi-Criteria Decision-Making Approach: Design Tactics Perspective," *IEEE Access*, vol. 8, pp. 25543–25556, 2020, doi: 10.1109/ACCESS.2020.2970784.

[48] N. Alhirabi, O. Rana, and C. Perera, "Security and Privacy Requirements for the Internet of Things: A Survey," *ACM Trans. Internet Things*, vol. 2, no. 1, Feb. 2021, doi: 10.1145/3437537.

[49] B. Tavares, M. Keil, C. Sanches, A. Diniz de Souza, and C. Silva, "A Risk Management Tool for Agile Software Development," vol. 1, p. 1, Dec. 2020, doi: 10.1080/08874417.2020.1839813.

[50] M. T. Baldassarre, V. S. Barletta, D. Caivano, and M. Scalera, "Integrating security and privacy in software development," *Softw. Qual. J.*, vol. 28, no. 3, pp. 987–1018, Sep. 2020, doi: 10.1007/s11219-020-09501-6.

[51] W. Williams, *Creating an Information Security Program from Scratch*, 1st ed. CRC Press, 2022.

[52] A. Ali, Y. Hafeez, S. Hussain, and S. Yang, "Role of Requirement Prioritization Technique to Improve the Quality of Highly-Configurable Systems," *IEEE Access*, vol. 8, pp. 27549–27573, 2020, doi: 10.1109/ACCESS.2020.2971382.

[53] L. V Astakhova, "Transformation of Strategic Models for Managing Human Risks of Information Security of an Enterprise as an Imperative of the Digital Industry," *Sci. Tech. Inf. Process.*, vol. 48, no. 2, pp. 71–77, Apr. 2021, doi: 10.3103/S0147688221020027.

[54] M. Alenezi and S. Almuairfi, "Security Risks in the Software Development Lifecycle," *Int. J. Recent Technol. Eng.*, vol. 8, pp. 7048–7055, Sep. 2019, doi: 10.35940/ijrte.C5374.098319.

[55] X. Chen and Y. Deng, "An Evidential Software Risk Evaluation Model," *Mathematics*, vol. 10, p. 2325, Jul. 2022, doi: 10.3390/math10132325.

[56] A. Goutam and V. Tiwari, "Vulnerability Assessment and Penetration Testing to Enhance the Security of Web Application," in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, 2019, pp. 601–605, doi: 10.1109/ISCON47742.2019.9036175.

[57] L. Gonchar, "Implementation of Secure Software Development Lifecycle in a Large Software Development Organization BT - Proceedings of the 21st International Workshop on Computer Science and Information Technologies (CSIT 2019)," Dec. 2019, pp. 137–139, doi: 10.2991/csit-19.2019.23.

[58] R. J. Curts and D. E. Campbell, *Building A Global Information Assurance Program*, 1st ed. New York: Auerbach Publications, 2003.

[59] H. O. Nwaete, "Secure Software Development: Industrial Practice - A Review," *i-Manager's J. Softw. Eng. Nagercoil*, vol. 16, no. 3, pp. 60–71, 2022, doi: 10.26634/jse.16.3.18674.

[60] E. Venson, X. Guo, Z. Yan, and B. Boehm, "Costing Secure Software Development: A Systematic Mapping Study," 2019, doi: 10.1145/3339252.3339263.

[61] A. A. Alghamdi and M. Niazi, "Challenges of Secure Software Deployment: An Empirical Study," in *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, 2022, pp. 440–445, doi: 10.1145/3530019.3531337.

[62] K. Rindell, S. Hyrynsalmi, and V. Leppänen, "Aligning security objectives with agile software development," 2018, doi: 10.1145/3234152.3234187.

[63] A. Petrikoglou and T. Kaskalis, *Full Stack Web Development Teaching: Current Status and a New Proposal*. 2019.

[64] D.-R. Khan, S. U. Khan, M. Ilyas, and H. Khan, "Systematic Mapping Study on Security Approaches in Secure Software Engineering," *IEEE Access*, vol. 9, pp. 19139–19159, Jan. 2021, doi: 10.1109/ACCESS.2021.3052311.

[65] Z. Stefanovska, K. Jakimoski, and W. Stefanovski, "Optimization of Secure Coding Practices in SDLC as Part of Cybersecurity Framework," *J. Comput. Sci. Res.*, vol. 4, Apr. 2022, doi: 10.30564/jcsr.v4i2.4048.

[66] Z. Shen and S. Chen, "A Survey of Automatic Software Vulnerability Detection, Program Repair, and Defect Prediction Techniques," *Secur. Commun. Networks*, vol. 2020, p. 8858010, 2020, doi: 10.1155/2020/8858010.

[67] P. Mukherjee and C. Mazumdar, "'Security Concern' as a Metric for Enterprise Business Processes," *IEEE Syst. J.*, vol. 13, no. 4, pp. 4015–4026, 2019, doi: 10.1109/JSYST.2019.2918116.

[68] F. H. Semantha, S. Azam, B. Shanmugam, K. C. Yeo, and A. R. Beeravolu, "A Conceptual Framework to Ensure Privacy in Patient Record Management System," *IEEE Access*, vol. 9, pp. 165667–165689, 2021, doi: 10.1109/ACCESS.2021.3134873.

[69] A. Pereira-Vale, G. Márquez, H. Astudillo, and E. B. Fernandez, "Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping," in *2019 XLV Latin American Computing Conference (CLEI)*, 2019, pp. 1–10, doi: 10.1109/CLEI47609.2019.235060.

[70] "ICST 2020 TOC," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. i–vii, doi: 10.1109/ICST46399.2020.00004.

[71] M. Howard and S. Lipner, *The Security Development Lifecycle*, vol. 34. 2006.

[72] T. Lorünser, H. Pöhls, L. Becker, and T. Laenger, *CryptSDLC: Embedding Cryptographic Engineering into Secure Software Development Lifecycle*. 2018.

[73] R. Brasoveanu, Y. Karabulut, and I. Pashchenko, "Security Maturity Self-Assessment Framework for Software Development Lifecycle," 2022, doi: 10.1145/3538969.3543806.

[74] N. Onumah, S. Attwood, and R. Kharel, "Towards Secure Application Development: A Cyber Security Centred Holistic Approach," in *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2020, pp. 1–6, doi: 10.1109/CSNDSP49049.2020.9249631.

[75] M. Alawneh and I. M. Abbadi, "Integrating Trusted Computing Mechanisms with Trust Models to Achieve Zero Trust Principles," in *2022 9th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, 2022, pp. 1–6, doi: 10.1109/IOTSMS58070.2022.10062269.

[76] R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, "Systematic Literature Review on Security Risks and its Practices in Secure Software Development," *IEEE Access*, vol. 10, pp. 5456–5481, 2022, doi: 10.1109/ACCESS.2022.3140181.

[77] M. M. Jakeri and M. F. Hassan, "A Review of Factors Influencing the Implementation of Secure Framework for in-House Web Application Development in Malaysian Public Sector," *2018 IEEE Conf. Appl. Inf. Netw. Secur.*, pp. 99–104, 2018, [Online]. Available: https://api.semanticscholar.org/CorpusID:59600770.

[78] T. Hanauer, W. Hommel, S. Metzger, and D. Pöhn, "A Process Framework for StakStakeholder-Specificualization of Security Metrics," 2018, doi: 10.1145/3230833.3232855.

[79] W. Williams, "Chapter Integrating Security into Software Development," in *Creating an Information Security Program from Scratch*, Ist., CRC Press, 2021, p. 222.

