Computers, Materials & Continua
DOI:10.32604/cmc.2022.019289
Article

Tech Science Press

Security Threat and Vulnerability Assessment and Measurement in Secure Software Development

Mamoona Humayun[1], NZ Jhanjhi[2,*], Maram Fahhad Almufareh[1] and Muhammad Ibrahim Khalil[3]

[1]Department of Information Systems, College of Computer and Information Sciences, Jouf University, Al-Jouf, KSA
[2]School of Computer Science and Engineering (SCE), Taylor's University, Selangor, Malaysia
[3]Department of Computer Science, Bahria University, Islamabad, Pakistan
*Corresponding Author: NZ Jhanjhi. Email: NoorZaman.Jhanjhi@taylors.edu.my
Received: 08 April 2021; Accepted: 10 May 2021

Abstract: Security is critical to the success of software, particularly in today's fast-paced, technology-driven environment. It ensures that data, code, and services maintain their CIA (Confidentiality, Integrity, and Availability). This is only possible if security is taken into account at all stages of the SDLC (Software Development Life Cycle). Various approaches to software quality have been developed, such as CMMI (Capability maturity model integration). However, there exists no explicit solution for incorporating security into all phases of SDLC. One of the major causes of pervasive vulnerabilities is a failure to prioritize security. Even the most proactive companies use the "patch and penetrate" strategy, in which security is accessed once the job is completed. Increased cost, time overrun, not integrating testing and input in SDLC, usage of third-party tools and components, and lack of knowledge are all reasons for not paying attention to the security angle during the SDLC, despite the fact that secure software development is essential for business continuity and survival in today's ICT world. There is a need to implement best practices in SDLC to address security at all levels. To fill this gap, we have provided a detailed overview of secure software development practices while taking care of project costs and deadlines. We proposed a secure SDLC framework based on the identified practices, which integrates the best security practices in various SDLC phases. A mathematical model is used to validate the proposed framework. A case study and findings show that the proposed system aids in the integration of security best practices into the overall SDLC, resulting in more secure applications.

Keywords: Security; secure software development; software development life cycle (SDLC); confidentiality; integrity; availability

1 Introduction

Software security is a process that includes the design, development, and testing of software for security where vulnerabilities are detected and revealed by the software itself [1–3]. It fundamentally

requires a proactive approach that takes place within the pre-deployment process. It's all about making the software development team do a great job to make it easier for operators. A simple error sometimes can end up causing millions of dollars of losses in today's business processes. But unfortunately, many software development companies do not follow best practices to incorporate security in SDLC [4,5]. This negligence includes lack of awareness, fear of time and cost overrun, use of third-party components and, lack of qualified professionals, etc. Due to the popularity and excessive usage of internet applications such as the internet of things, cloud computing, social media systems, etc., the number of security vulnerabilities is also overgrowing [6–9].

SDLC is a mechanism that generates the best quality and low-cost software in the shortest possible time. It offers a well-structured step flow that helps an enterprise easily produce high-quality, well-tested, and ready-to-use software. The common phases of SLDC include planning, analysis, design, implementation, testing & integration, and maintenance [10–12]. All these phases are dependent on each other and are of equal importance. If security is incorporated during all phases of SDLC then the resultant product will not be vulnerable to security threats. This is only possible if a secure SDLC process is followed, secure SDLC ensures that security-related activities are an integral part of the overall development effort [13–16].

Traditional security mechanisms mainly focus on network systems, and they spent a huge amount of money to make their network secure. These mechanisms include IDs (Intrusion detection system), firewalls, encryption, antivirus, and antispyware [17–19]. Further, security is considered an afterthought that is usually addressed after completing the development cycle using the approach of P&P (penetrate and patch), which means creating the patches for the available flaws. The drawback of the P&P technique is that the application users do not apply these patches. Further, attackers might plan and penetrate new vulnerabilities [20–23]. According to a report published by RiskIQ, security vulnerabilities alone cost as much as $25 per minute to the major companies while crypto companies face the loss of almost $2000 per minute due to cybercrimes [24]. Another report presented by positive technologies; 9 out of 10 web applications are vulnerable to security threats and about 39% of websites are vulnerable to unauthorized access, while data breach is a threat for about 64% of applications [25]. According to this report, 82 percent of vulnerabilities were due to flaws in code. This report has also published the severity of web application vulnerabilities in the past. This shows that security is one of the serious issues in the current era that need to be addressed carefully during SDLC. Further, the relative cost of addressing bugs and failure increase as the project progress as mentioned in the IBM system science institute report [26]. Therefore, handling security from the beginning of the project is necessary to save the software from future security breaches.

It is evident from the above discussion that secure software development is inevitable for improving project quality and reducing bug fixing cost, and there exists no explicit solution to this problem. As a contribution to research, we reviewed the current literature on vulnerability evaluation and assessment in SDLC and outlined the security best practices to evaluate and quantify security threats and vulnerabilities in SDLC. Based on the identified best practices, we have proposed a secure SDLC framework. The proposed framework tries to mitigate the security vulnerabilities in SDLC by addressing end-to-end security. The proposed framework is validated using a mathematical model and a case study. Fig. 1 illustrates the research process that was followed to carry out this research study.

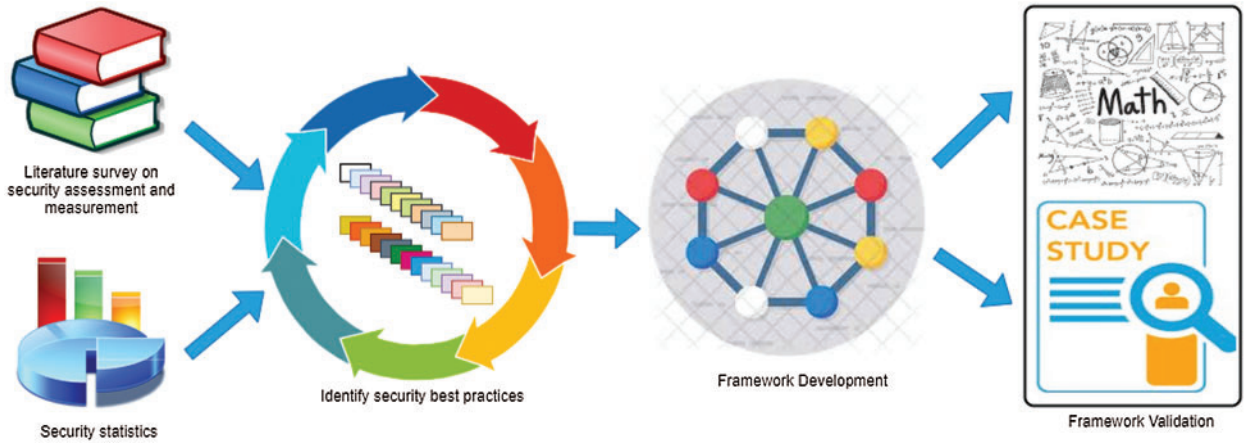

Figure 1: Research process

The results of the case study show that the proposed approach helps incorporate security in SDLC. The remaining paper is structured as; Section 2 describes the existing work to highlight the best practices and techniques available for the assessment and measurements of security threats and vulnerabilities. Section 3 presents our proposed framework that addresses security in SDLC. Sections 4 & 5 evaluate the proposed framework using mathematical modeling and case study. Section 6 discusses the findings of the study. Section 7 concludes this paper by providing directions for future work.

2 Literature Review

This section will provide an overview of some latest research in secure software development to highlight the best practices that need to be followed for developing quality software. Further, it will pave the way for the proposed framework.

In paper [27], an integrated security framework is proposed for secure SDLC. Security test cases and guidelines were generated based on the security activities, and best practices followed in secure SDLC. Security testing tools were integrated for the automation of test case execution. A prototype was constructed to evaluate the proposed framework. The results of the experiment showed that the proposed approach provides stable service with enhanced quality and security. In Paper [20], a Multi-vocal literature review was conducted to identify the best practices for designing secure software. Based on identified best practices, a framework Secure Software Design Maturity Model (SSDMM) was developed. The framework was evaluated using case studies, and the results show that SSDMM helps measure the maturity level of an organization. Further, SSDMMM helps organizations in the evaluation and improvement of software design security practices.

In paper [28], a systematic literature review (SLR) was performed to pinpoint the required practices for developing secure software. This paper also amended Somerville's requirement engineering practices. After identifying best requirement practices, a framework for secure requirement engineering named as Requirements Engineering Security Maturity Model (RESMM) was developed. The proposed framework was tested using questionnaires and case studies. The results show that the proposed framework is useful and easily adaptable. According to [29], security is not considered in the overall SDLC due to which a lot of security breaches occur. This paper presents a secure paradigm that is an extension of security development practices in agile methodology to overcome this problem in web application development. The proposed paradigm consists of three phases namely, inception,

construction, and transition. Further, this paper classifies security vulnerabilities and common risks and threats that occur during web application development. Based on identified gaps, a framework is proposed for secure web application development. The survey method was used for the evaluation of the proposed framework, and the results were satisfactory.

According to the paper [30], the use of best practices for risk management should be followed in overall SDLC for getting a quality software product. This paper discussed various practices of risk management and security in different phases of SDLC. It provides an insight to the researchers and practitioners about the existing best practices that need to be followed. According to [31], security is an important aspect of software systems. However, existing studies do not address it explicitly into SDLC, therefore, this study identified important security policies, practices, and tools within SDLC and proposed a model for incorporating these elements into SDLC. This research study used a case study-based approach for answering research questions. Further, an expert review was conducted for the validation of the proposed model.

Paper [32] concludes that understanding software and proper application methods results in a reliable and quality software product. This research identified the issues that occurred while incorporating security into SDLC along with suitable solutions. Further, it discussed some security-related issues in detail such as security testing, threat modeling, risk assessment, and other suitable techniques that help in developing secure software. Paper [33] investigates security aspects in various phases of SDLC and evaluates these aspects with the help of the research community and software engineers. The results obtained from this qualitative study were analyzed using the SPSS tool, some security rules were also proposed for various phases of SDLC.

According to [34], security has been considered an afterthought for a long time. However, this approach is not suitable in today's fast-paced economy. Security needs to be incorporated from the beginning of software till the end. This paper provides an overview of security plans in various phases of SDLC. Further, it emphasizes the importance of good governance for the success of the project. A systematic mapping study is performed in [35] to identify the existing security approaches, followed in SDLC. In this paper, 118 studies were selected as the primary studies, and 52 security practices were identified from the selected studies. According to study findings, most of the security practices are being followed in the coding stage of SDLC.

It is obvious from the above discussion that incorporating security in different phases of SDLC is inevitable for quality software. There exist various studies that discuss the importance of incorporating security in SDLC, however, still there exists space for further research in the area. As a contribution towards this research direction, first, we have highlighted the common reasons for security flaws in SDLC as shown in Fig. 2. by providing a taxonomy of SDLC [36–42]. Next, we have proposed an approach that will incorporate security best practices in various phases of SDLC as mentioned in the upcoming section. Further, we have proposed a secure SDLC framework in the next section that is evaluated using a mathematical model and case study.

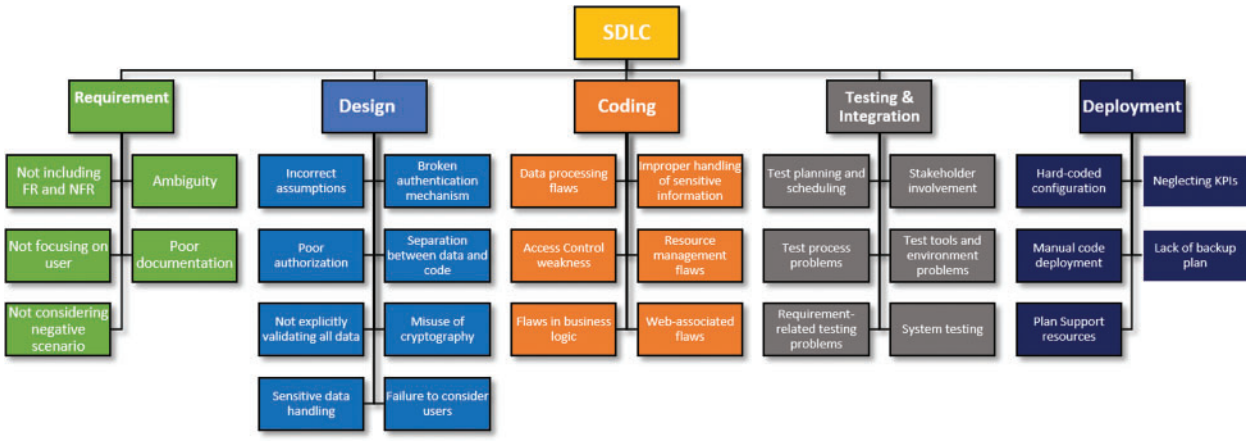

Figure 2: A taxonomy of security flaws reasons in SDLC

3 Proposed Framework

This section will discuss different phases of SDLC, the discussion of each phase will focus on three dimensions; the tasks performed in that phase, security issues involved, and mitigation strategies. It will not only provide a detailed overview of secure SDLC rather will also pave the way for our proposed solution. Below we discuss these phases briefly.

3.1 Requirement Phase

Requirement engineering is the first phase of SDLC, and the success of this phase leads towards a better software product. Further, handling security from the requirement phase help to save rework and additional cost. The tasks performed at this level are listed in Column 1 of Tab. 1 [43–45]. Existing literature on requirement security has highlighted different issues that might occur if security is not incorporated from the beginning. Some common security issues that might occur during the requirement phase of SDLC are listed in column 2 of Tab. 1 [30,33,46,47]. To accomplish this phase and to address security from the beginning, best practices need to be followed. Different researchers have proposed different practices that need to be followed. Column 3 of Tab. 1 list down the commonly used best practices for handling security during the requirement phase of SDLC [28,30,33,48–50]

Table 1: Requirement phase activities/issues and solutions

Activities	Issues/challenges	Solutions
+Inception	+Shared understanding of requirements	+All stakeholders need to be agreed on requirement definitions.
+Elicitation	+Elicitation of security requirements	+Identify critical and vulnerable assets.
+Elaboration	+Lack of defense in depth	+ Identify requirement dependencies.
+Negotiation	+Lack of security awareness	+ Identify threat & possible risks
+Specification		+Develop corresponding artifacts.

(Continued)

Table 1: Continued

Activities	Issues/challenges	Solutions
+Validation +Management		+Elicit security requirements. +Perform requirement prioritization & classification. +Perform requirement inspection. +Update requirement repository

3.2 Design Phase

Design is an essential step of the SDLC because it determines the look and sound of the app. Furthermore, it offers a user-interactive platform, making it vulnerable to numerous security threats. The important tasks performed during this phase are listed in Column 1 of Tab. 2 [51–55]. Common security issues that are usually faced during software design are listed in Column 2 of Tab. 2 [20,53,56–59] while security best practices are listed in Column 3 of Tab. 2 [13,14,20,28,52,53,60,61]

Table 2: Design phase activities/issues and solutions

Activities	Issues/challenges	Solutions
+Identify design assets.	+Establishing design security requirements.	+Apply economy of mechanism policy to keep your design as simple as you can.
+Abstract specifications	+Evaluating security risks of third-party components.	+Apply false-safe default principles to make sure that the failure of any activity will prevent unsafe operation.
+Architectural design	+Traceability	+Apply access control mechanism to make sure that every object is checked for authorization.
+Component design	+Access control	+Give least privileges to save the system from security attacks.
+Interface design	+Lack of defense in depth	+Follow least common mechanism to restrict shared resource access.
+Database design	+Lack of security awareness	+Follow psychological acceptability principle of design to automatically incorporate basic security.

(Continued)

Table 2: Continued

Activities	Issues/challenges	Solutions
	+Design flaws	+Apply Defense-in-depth policy which includes multilevel security. +Design review should be performed to validate the design.

3.3 Coding Phase

The practice of secure coding is inevitable for safeguarding computer software against security vulnerabilities; therefore, the coding phase is among the critical phases of SDLC. Tasks performed during this phase are listed in Column 1 of Tab. 3 The selection of appropriate coding language and classification of modules is a challenging task. Further, the reusability of code also creates a challenge if security is not considered while coding. Column 2 of Tab. 3 list down the security issues that are usually faced during the coding phase while best practices of secure coding are listed in Column 3 of Tab. 3 [28,33,62–70]

Table 3: Coding phase activities/issues and solutions

Activities	Issues/challenges	Solutions
+Choose coding language.	+Buffer overflow	+Perform secure coding by following a secure coding checklist and practices.
+Modules classification	+Code injection flaw	+Follow OWASP secure coding practices and checklists.
+Choose programming tools.	+Lack of using secure coding practices.	+Follow OWASP general coding practices.
+Consider reusability options.	+Lack of security awareness	+Perform pair programming if possible.
	+Evaluating security risks of third-party components.	

3.4 Testing & Integration Phase

The testing & integration phase aims to make sure that all the system components provide their required functionality alone and as part of the whole system. The tasks involved in this phase are listed in colum1 of Tab. 4. This phase aims to find possible bugs and errors in the system and remove them. Some common security issues involved in this phase are listed in Column 2 of Tab. 4. This phase gives the final touch to the software before deployment therefore quality must be assured. Column 3 of Tab. 4 lists down the best practices that help to make this phase secure and successful [71–77].

Table 4: Testing & integration phase activities/issues and solutions

Activities	Issues/challenges	Solutions
+Initiate testing activities	+Tools selection	+Perform secure testing and integration.
+System testing.	+Using multiple approaches.	+Test cases should be generated based on the output of the requirement phase.
+Security testing	+Developer acceptance/resistance	+Perform functional testing.
+Acceptance testing	+Compliance issues	+Perform nonfunctional testing.
+Integration testing	+Time/budget constraints +Traceability +Technical risks	+Perform integration testing.

3.5 Deployment Phase

This is the last stage of SDLC which handles the release and change management. In this phase, the software is installed in its actual environment. It seems simple but pairing the software with the existing environment is sometimes complex. Patches are created to handle the flaws; this makes the software vulnerable to various security threats. Column 1 of Tab. 5 lists down the tasks that are performed in this stage. Some common security issues involved in this stage are listed in Column 2 of Tab. 5. Further, customer satisfaction is very important at this level therefore Column 3 of Tab. 5 list down the best practices that need to be considered for making this phase successful [78–83].

Table 5: Deployment phase activities/issues and solutions

Activities	Issues/challenges	Solutions
+Corrective maintenance	+possibility of misconfiguration	+document change management process
+Adaptive maintenance	+possibility of flaws	+follow change management process.
+Perfective maintenance +Preventive maintenance		+Plan support resources

The above discussion has highlighted the brief details of SDLC phases along with security issues and mitigation strategies. Based on identified challenges and best practices for each phase of SDLC, we have developed a framework as shown in Fig. 3. This framework addresses security in overall SDLC by incorporating security best practices in different phases of SDLC. The framework is divided into two dimensions. The horizontal dimension of the framework shows SDLC phases while the vertical dimension of the framework list down the details of the tasks performed during each phase of SDLC, security issues involved in each phase, and corresponding mitigation strategies.

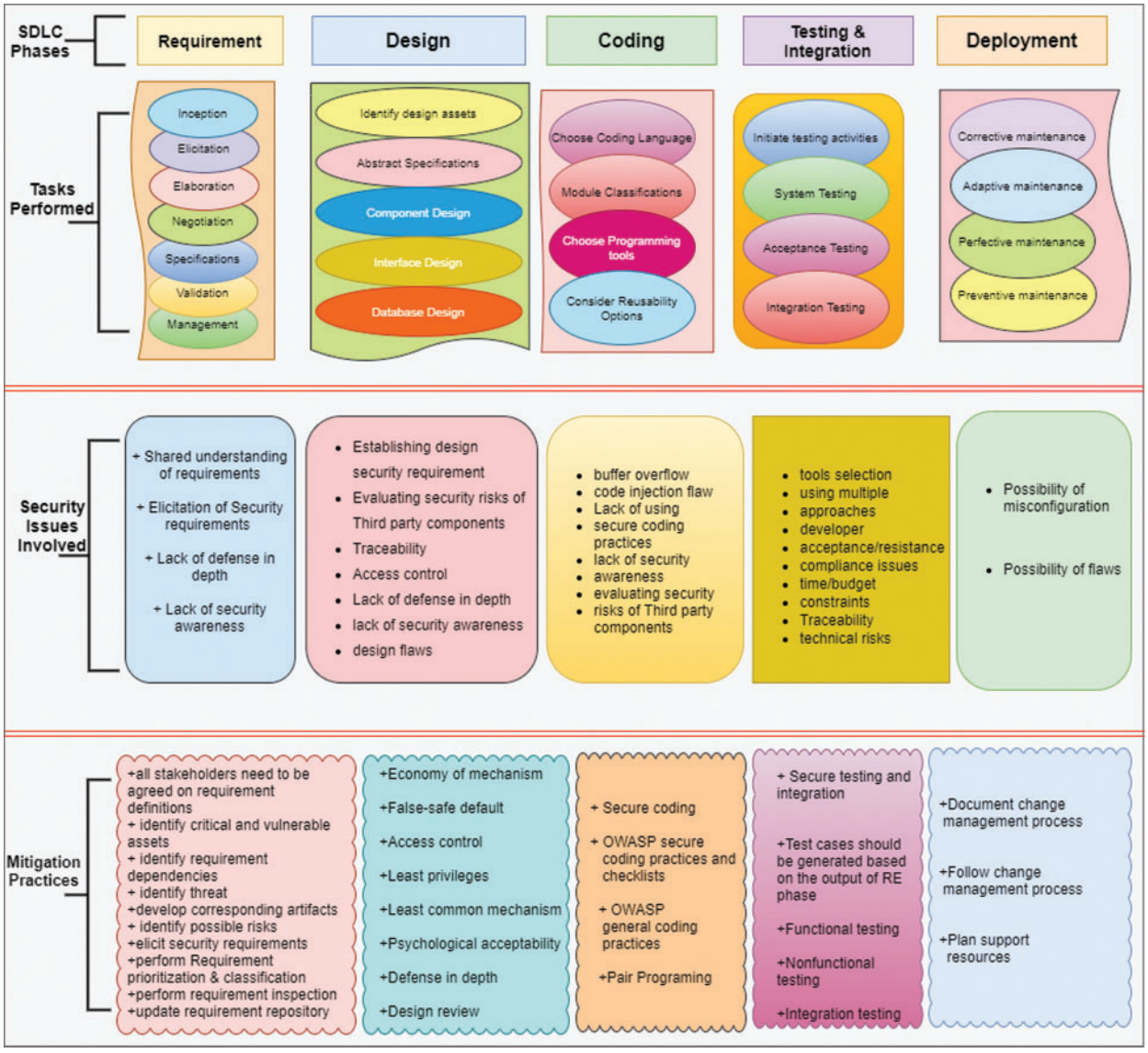

Figure 3: Proposed secure SDLC framework

According to the proposed framework, security best practices need to be incorporated from the beginning of the project until deployment to get secure and quality software. The practices mentioned in the above framework will not only make the software secure but will also not add much to the project budget and time. The detail of the framework is also presented in the form of an algorithm.

Algorithm 1: Algorithm for Proposed Approach

Let R = Requirements, D = Design, C = Coding/implementation, T&I = testing and integration, M = maintenance, T = Threat, ART = artifacts

1. Begin

(Continued)

Algorithm 1: Continued

2. *Use SREP( )* // follow Secure Requirement Engineering Process (SREP)
   a. *Agree on R* // all stakeholders need to be agreed on requirement definitions
   b. *Identify CVA* // identify Critical and Vulnerable Assets (CVA)
   c. *Identify RD* // identify Requirement Dependencies (RD)
   d. *Identify T* // identify threat
   e. *develop ART* // develop corresponding artifacts
   f. *Identify RS* // identify possible Risks (RS)
   g. *Elicit SR* // elicit Security Requirements (SR)
   h. *Perform RPC* // perform Requirement Prioritization & Classification (RPC)
   i. *Perform RI* // perform Requirement Inspection (RI)
   j. *Update RR* // update Requirement Repository (RR)
3. If Step a to j are successful then Go to 5
4. Else Go to 2
5. Perform SSD () // follow Secure Software Design (SSD)
   a. *Follow EOM* // Apply Economy of Mechanism (EOM) and keep your design as simple as you can
   b. *Apply FSD* // Apply False-Safe Default (FSD) principles to make sure that failure of any activity will prevent unsafe operation
   c. *Apply ACM* // Apply Access Control Mechanism (ACM) to make sure that every object is checked for authorization
   d. *Give LP* // give Least Privileges LP)
   e. *Follow LCM* // Least Common Mechanism (LCM) to restrict shared resource access
   f. *Ensure PA* // Psychological Acceptability (PA) of design automatically incorporate basic security
   g. *Apply DID* // Defense in Depth (DID) include multilevel security
   h. *Perform DR* // Design Review (DR) should be performed to validate design

6 f Step a to h are successful then Go to 8

7. Else Go to 5
8. Perform SC () // perform Secure Coding (SC) by following secure coding checklist and practices
   a. *Follow SCP* // follow OWASP Secure Coding Practices (SCP) and checklists
   b. *Follow GCP* // follow OWASP General Coding Practices (GCP)
   c. *Perform PP* // perform Pair Programing (PP) if possible
9. If Step a to c are successful then Go to 11
10. Else Go to 8
11. Perform ST&I () // Perform Secure Testing and Integration(ST&I)
   a. *Generate TC* // Test Cases (TC) should be generated based on the output of Step 2
   b. *Perform FT* // perform Functional Testing (FT)
   c. *Perform NFT* // perform Nonfunctional Testing (NFT)
   d. *Perform IT* // perform Integration Testing (IT)

(Continued)

Algorithm 1: Continued

12. If Step a to d are successful then Go to 14
13. Else Go to 11
14. Perform M()
    a. *Document CMP* // documents Change Management Process (CMP)
    b. *Follow CMP* // follow change management process
    c. *Plan SR* // Plan Support Resources (SR)
15. If Step a to c are successful then Go to 17
16. Else Go to 14
17. END

4 Framework Evaluation Using Mathematical Modeling

Before proceeding towards mathematical modeling, we first define notations used in the mathematical model for getting a better understanding. Tab. 6 lists down the notations of our mathematical model.

Table 6: Notations sued in the mathematical model

Symbol	Used for	Symbol	Used for
S	SDLC	Dp	Deployment
R	Requirement phase	Sc	Security
D	Design phase	TV	Threshold value
C	Coding Phase	OF	Objective function
TI	Testing & integration		

According to the proposed framework, secure SDLC need to incorporate security in overall SDLC as shown in Eq. (1)

$$S = f(R, D, C, TI, Dp) \tag{1}$$

The proposed framework aims to enhance security, therefore, the objective function in our case will be as shown in Eq. (2)

$$OF = Max(Sc) \tag{2}$$

The Requirement phase of SDLC can be improved by following best practices as mentioned in the proposed framework. Hence the requirement phase can be modeled as

$$R = \sum_{i=1}^{n} X_i \tag{3}$$

where $X_1, X_2 \ldots X_n$ is a set of best practices that need to be followed by organizations for making the requirement phase secure. To measure the security of this phase, we need to assign a weight to each $X_1, X_2 \ldots X_n$ as $W_1, W_2 \ldots W_n$ and set a threshold value for R. In this case, the measurement of R will be done using the formula in Eq. (4) as follows.

$$1/n(X_1W_1 + X_2W_2 + X_3W_3 \ldots + X_nW_{n)} \geq Tv(R) \tag{4}$$

Eq. (4) must be true for secure completion of the requirement phase. The value of Tv varies from project to project and will be determined based on the project's nature. In the same way, secure design can be achieved by following the best design practices as shown in Eq. (5)

$$D = \sum_{j=1}^{n} Y_j \tag{5}$$

where Y_1 , Y_2 , $Y_3 \ldots Y_n$ are best practices that need to be followed for making software design secure? To measure the security of software design, the organization need to set the value of Tv based on the nature of the project and assign weights to each Y_1 , Y_2 , $Y_3 \ldots Y_n$. Then the measurement of design security will be done using the formula mentioned in Eq. (6)

$$1/n(Y_1W_1 + Y_2W_2 + Y_3W_3 \ldots + Y_nW_n) \geq Tv(D) \tag{6}$$

Eq. (6) must be true for the completion of the secure software design phase. Once the requirement and design phase is complete, the organization moves towards the coding phase. Whatsoever the development model an organization is following, the basic activities/phases of SDLC remain almost the same. To make the coding process secure, organizations need to follow the best security practices during coding as mentioned in Eq. (7).

$$C = \sum_{k=1}^{n} Z_k \tag{7}$$

where Z_1 , Z_2 , $Z_3 \ldots Z_n$ are the secure coding practices that need to be considered during the coding phase of SDLC. The security of the coding phase will be measured by assigning weights to each Z_1 , Z_2 , $Z_3 \ldots Z_n$ according to their priority in the project. The formula in Eq. (8) will be used to measure the security of the coding phase by setting the value of Tv according to the organization's preferences.

$$1/n(Z_1W_1 + Z_2W_2 + Z_3W_3 \ldots + Z_nW_n) \geq Tv(C) \tag{8}$$

The weighted total of coding practices must be greater than the set threshold value for secure coding. Once the coding is done securely, the software team moves towards the testing and integration phase. This phase is critical as all the bugs and errors must be removed during this phase otherwise software will be handover to the customer after the completion of this phase. Customer acceptability is inevitable for business continuity and project acceptance therefore security of the testing and integration phase must be ensured. The organizations need to follow security best practices during this phase as shown in Eq. (9).

$$TI = \sum_{l=1}^{n} U_l \tag{9}$$

where U_1 , U_2 , $U_3 \ldots U_n$ are the security best practices that need to be incorporated in the testing and integration phase of SDLC for making it secure. To measure the security of this phase, the formula in Eq. (10) will be used.

$$1/n(U_1W_1 + U_2W_2 + U_3W_3 \ldots + U_nW_n) \geq Tv(C) \tag{10}$$

The weighted total of best testing and integration practices should result in more than the set threshold value for ensuring the security of this phase.

The last phase of SDLC is a deployment where software is installed in its working environment, any change requested by the user is accommodated at this stage. This phase should be planned carefully to avoid any inconsistency and dissatisfaction from the user. According to the proposed model, a set of best practices need to be followed during this phase to make it secure and satisfactory. Eq. (11) illustrates the best practices of the deployment phase.

$$Dp = \sum_{m=1}^{n} V_m \tag{11}$$

where V_1 , V_2 , $V_3 \ldots V_n$ are the best practices that need to be considered during the deployment phase of SDLC for making it secure. To measure the security of this phase, the formula in Eq. (12) will be used.

$$1/n(V_1W_1 + V_2W_2 + V_3W_3 \ldots + V_nW_n) \geq Tv(Dp) \tag{12}$$

The weighted total of the best practices incorporated in the deployment phase must be greater than the set value of the threshold for the secure execution of this phase.

Once all the phases of SDLC are done while considering security as a priority and incorporating security best practices in SDLC. The organizations need to check the overall security of the project against a threshold value that is set for the project based on its nature. The accumulative security can be calculated using the formula of Eq. (13)

$$Acc\ (Sc) = 1/n \sum_{i=1}^{n} X_iW_i + 1/n \sum_{j=1}^{n} Y_jW_j + 1/n \sum_{k=1}^{n} Z_kW_k + 1/n \sum_{l=1}^{n} U_lW_l + 1/n \sum_{m=1}^{n} V_mW_m \tag{13}$$

where accumulative security must be greater than the set value of the accumulative threshold as shown in Eq. (14)

$$Acc(Sc) \geq Acc(Tv) \tag{14}$$

Once the organization achieves accumulative security for its developed software by incorporating the best practices mentioned in the proposed framework, the resultant software is resistant against security vulnerabilities and threats.

5 Framework Evaluation Using Case Study

Security is inevitable for all kinds of software projects; however, it varies from project to project. Some systems are security-critical as compared to others. The traditional security mechanism of P&P is sometimes more costly and complex. Therefore, security must be incorporated in the overall SDLC. There exist various approaches for integrating security into SDLC but still, the problem persists. To handle security in overall SDLC, we have proposed a secure SDLC framework. According to the proposed framework, Security must be incorporated from the beginning until the software is deployed in its working environment.

The organization XYZ follows the proposed framework, it set the values for different parameters according to the project's nature. The concept mapping technique is used to measure the actual values

of best practices used in various phases of SDLC. In this technique, important concepts related to the practice are identified and a panel of a software team is asked to map these concepts based on their understanding. This technique is very useful for measuring qualitative attributes [84–86]. The last column of Tab. 7 shows the obtained value for each security best practice using the concept mapping technique. Column 3 of Tab. 7 shows the weight for each practice that was decided based on the nature of the project and the importance of that practice for the project's security.

Table 7: Data used for cases study

P	Practice definition	W	OV
	SDLC requirement phase security		
P1	All stakeholders need to be agreed on requirement definitions	0.80	0.85
P2	Identify critical and vulnerable assets	0.90	0.88
P3	Identify requirement dependencies	0.95	0.90
P4	Identify threat	0.90	0.88
P5	Develop corresponding artifacts	0.90	0.95
P6	Identify possible risks	0.90	0.90
P7	Elicit security requirements	0.95	0.80
P8	Perform Requirement prioritization & classification	0.90	0.85
P9	Perform requirement inspection	0.90	0.80
P1	Update requirement repository	0.90	0.85
	SDLC design phase security		
P1	Apply economy of mechanism policy to keep your design as simple as you can	0.90	0.85
P2	Apply false-safe default principles to make sure that failure of any activity will prevent unsafe operation	0.90	0.80
P3	Apply access control mechanism to make sure that every object is checked for authorization	0.85	0.80
P4	Give least privileges to save the system from security attacks	0.80	0.75
P5	Follow least common mechanism to restrict shared resource access	0.80	0.70
P6	Follow psychological acceptability principle of design to automatically incorporate basic security	0.80	0.85
P7	Apply defense in depth policy which include multilevel of security	0.90	0.90
P8	Design review should be performed to validate design	0.90	0.90
	SDLC coding phase security		
P1	Perform secure coding by following secure coding checklist and practices	0.90	0.80
P2	Follow OWASP secure coding practices and checklists	0.80	0.70

(Continued)

Table 7: Continued

P	Practice definition	W	OV
	SDLC coding phase security		
P3	Follow OWASP general coding practices	0.80	0.75
P4	Perform pair programing if possible	0.75	0.50
	SDLC Testing & integration phase security		
P1	Perform secure testing and integration	0.80	0.80
P2	Test cases should be generated based on the output of requirement phase	0.90	0.90
P3	Perform functional testing	0.90	0.95
P4	Perform nonfunctional testing	0.80	0.85
P5	Perform integration testing	0.90	0.95
	SDLC deployment phase security		
P1	Document change management process	0.90	0.80
P2	Follow change management process	0.85	0.80
P3	Plan support resources	0.80	0.80

Note: Where P = Security best practice, W = weight and OV = obtained value of security

$Sc(R) = 1/n(X_1W_1 + X_2W_2 + X_3W_3 \ldots + X_nW_n)$ Where n is the total number of best security practices used during the requirement phase. By substituting the values from Tab. 7 into the above equation we get

$$Sc(R) = 1/10 \left(\begin{array}{l} (0.85)(0.80) + (0.88)(0.90) + (0.90)(0.95) + (0.88)(0.90) + (0.95)(0.90) + \\ (0.90)(0.90) + (0.80)(0.95) + (0.85)(0.90) + (0.80)(0.90) + (0.85)(0.90) \end{array} \right)$$
$$= 7.794/10 = 0.7794$$

The total value of $Sc(R)$ is 0.7794 while the threshold value is 0.75 as shown in Tab. 8. This shows that incorporating the security best practices in the requirement phase of SDLC help to improve the security of this phase.

Table 8: Threshold values for SDLC phases

SDLC phases	Threshold values		
Security of requirement	0.75	Testing & integration security	0.60
Design security	0.68	Deployment security	0.50
Coding security	0.55	Accumulative security	0.60

Similarly, $Sc(D) = 1/n(Y_1W_1 + Y_2W_2 + Y_3W_3 \ldots + Y_nW_n)$ by substituting the obtained values for each security best practice during the design phase of SDLC into the above equation we get

$$Sc(D) = 1/8 \left(\begin{array}{l} (0.85)(0.90) + (0.80)(0.90) + (0.80)(0.85) + (0.75)(0.80) \\ +(0.70)(0.80) + (0.85)(0.80) + (0.90)(0.90) + (0.90)(0.90) \end{array} \right)$$
$$= 5.625/8 = 0.70$$

The obtained security value of $Sc(D)$ is 0.70 while the threshold value is 0.68. This also shows that the security best practices mentioned in the proposed framework help in improving the security of the design phase.

$Sc(C) = 1/n(Z_1W_1 + Z_2W_2 + Z_3W_3 \ldots + Z_nW_n)$ By substituting the obtained values for secure coding practices into the given equation we obtained results as

$$Sc(C) = 1/4((0.80)(0.90) + (0.70)(0.80) + (0.75)(0.80) + (0.5)(0.75)) = 2.255/5 = 0.56$$

The obtained security value for the coding phase is 0.56 while the threshold value is 0.55. This shows that organizations need to incorporate security best practices during the SDLC coding phase as mentioned in the proposed framework.

The security values of the testing & integration phase can be calculated by using the formula. $Sc(TI) = 1/n(W_1 + U_2W_2 + U_3W_3 \ldots + U_nW_n)$By substituting the values from Tab. 3 into the above equation we get

$$Sc(TI) = 1/5((0.80)(0.80) + (0.90)(0.90) + (0.95)(0.90) + (0.85)(0.80) + (0.95)(0.90)) = 3.84/5 = 0.768$$

The security value for testing and integration phase obtained after applying security best practices as mentioned in the proposed framework is 0.768 which is higher than the threshold value of 0.60. This shows that the proposed framework helps improve the security of testing & integration phase of SDLC.$Sc(Dp) = 1/n(V_1W_1 + V_2W_2 + V_3W_3 \ldots + V_nW_n)$ by substituting the obtained values in the equation, we get

$$= 1/3((0.80)(0.90) + (0.80)(0.85) + (0.80)(0.80)) = 2.04/3 = 0.68$$

The obtained security value for the deployment phase is 0.68 which is greater than the threshold value of 0.50 which shows that security best practices need to be incorporated in the SDLC deployment phase.

Now we find the accumulative security by using the formula below.

$$Acc\ (Sc) = \frac{1}{5}(1/n\sum_{i=1}^{n} X_iW_i + 1/n\sum_{j=1}^{n} Y_jW_j + 1/n\sum_{k=1}^{n} Z_kW_k + 1/n\sum_{l=1}^{n} U_lW_l + 1/n\sum_{m=1}^{m} V_mW_m)$$
$$= 1/5(0.7794 + 0.70 + 0.56 + 0.768 + 0.68) = 3.4874/5 \cong 0.70$$

The obtained value for accumulative security is 0.70 which is also greater than the threshold value for the cumulative security of 0.60. The results of the case study show that the proposed framework helps improve the security of SDLC.

6 Discussion

Security is one of the important factors that need to be considered from the very beginning of the software development process. Bugs and errors which are detected in the early phases of development are easy and cheap to handle as compared to the ones which are captured during later phases. Therefore, incorporating security in overall SDLC is inevitable for secure software development as well as organizations' business continuity and avoiding rework. Traditionally, security is considered an afterthought activity that is handled by creating patches for the flaws identified during testing of the project or after deployment. However, the P&P strategy is not easy to implement in today's software

development environment where billions of devices are interconnected, and software has to work as an integral part of the overall system.

To incorporate security in the overall development cycle, we have done a detailed literature review and identified the best practices that help manage security in SDLC. Based on the identified security best practices, we have formulated a secure SDLC framework. The structure of the framework that has been suggested is two-dimensional. The phases of SDLC are represented on the horizontal axis, while the vertical axis is separated into three layers; layer 1 highlights the critical tasks performed in the corresponding SDLC process. Layer 2 describes the security issues involved in the SDLC phase while best practices for overcoming the listed security problems are discussed in layer 3. The proposed framework was modeled mathematically and was also evaluated using a case study. The case study results show that incorporating security best practices in different phases of SDLC improve software security.

7 Conclusion and Future Work

One of the most critical things to be considered from the start of the software development process is security. When bugs and defects are discovered early in the production process, they are easier and less expensive to fix than those discovered later. In the past, many software failed due to negligence of the security factor. Testing the software for security after development is not only time-consuming and complex; rather, it increases the time and cost of the project. To avoid complexity and project failure at a later stage, it is necessary to consider security as an important attribute of the software from the beginning of the project until the deployment. To address this issue, we have provided a framework based on existing security best practices for different phases of SDLC. The proposed framework was evaluated using a mathematical model and a case study and results show that the proposed framework helps improve the security of SDLC.

In the future, we are planning to extend the proposed framework by incorporating more security best practices and evaluating it on a security-critical project.

Funding Statement: The authors received no specific funding for this study

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. F. Wen and B. Katt, "Learning software security in context: An evaluation in open source software development environment," in *Proc. ARES*, Canterbury, UK, pp. 1–10, 2019.

[2] M. Humayun, M. Niazi, N. Z. Jhanjhi, M. Alshayeb and S. Mahmood, "Cyber security threats and vulnerabilities: A systematic mapping study," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 1–19, 2020.

[3] S. Dotcenko, A. Vladyko and I. Letenko, "A fuzzy logic-based information security management for software-defined networks," in *Proc ICACT*, Phoenix Park, Pyeong Chang, Korea, pp. 167–171, 2014.

[4] W. Khreich, S. S. Murtaza, A. H. Alhadj and C. Talhi, "Combining heterogeneous anomaly detectors for improved software security," *Journal of Systems and Software*, vol. 137, no. 1, pp. 415–429, 2018.

[5] S. Hosseinzadeh, S. Rauti, S. Lauren, J. M. Makela and J. Holvitie *et al.,* "Diversification and obfuscation techniques for software security: A systematic literature review," *Information and Software Technology*, vol. 104, no. 1, pp. 72–93, 2018.

[6] D. K. Alferidah and N. Z. Jhanjhi, "A review on security and privacy issues and challenges in internet of things," *International Journal of Computer Science and Network Security*, vol. 20, no. 4, pp. 263–286, 2020.
[7] D. K. Alferidah and N. Jhanjhi, "Cybersecurity impact over big data and IoT growth, in *Proc ICCI*, Istanbul, Turkey, pp. 103–108, 2020.
[8] O. L. Ben, G. Chehrazi, E. Bodden, P. Tsalovski and A. D. Brucker, "Time for addressing software security issues: Prediction models and impacting factors," *Data Science and Engineering*, vol. 2, no. 2, pp. 107–124, 2017.
[9] B. Kim, "Open source software security issues and applying a secure coding scheme," *KIISE Transactions on Computing Practices*, vol. 23, no. 8, pp. 487–491, 2017.
[10] S. Z. Hlaing and K. Ochimizu, "An integrated cost-effective security requirement engineering process in SDLC using FRAM," in *Proc CSCI*, Las Vegas, Nevada, USA, pp. 852–857, 2018.
[11] B. Sugiantoro, M. Anshari and D. Sudrajat, "Developing framework for web based e-commerce: Secure-SDLC, *Journal of Physics: Conference Series*, vol. 1566, no. 1, pp. 012020, 2020.
[12] A. M. Fernandes, A. Pai and L. M. Colaco, "Secure SDLC for IoT based health monitor," in *Proc ICECA*, Coimbatore, India, pp. 1236–1241, 2018.
[13] R. Khaim, S. Naz, F. Abbas, N. Iqbal and M. Hamayun, "A review of security integration technique in agile software development," *International Journal of Software Engineering & Applications*, vol. 7, no. 3, pp. 49–68, 2016.
[14] D. A. Arrey, "Exploring the integration of security into Software Development Life Cycle (SDLC) methodology," Ph.D. dissertation, Colorado Technical University, Colorado, 2019.
[15] S. Kang and S. Kim, "CIA-Level driven secure SDLC framework for integrating security into SDLC process," *Journal of the Korea Institute of Information Security & Cryptology*, vol. 30, no. 5, pp. 909–928, 2020.
[16] T. Bhuvaneswari and S. Prabaharan, "A survey on software development life cycle models," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 5, pp. 262–267, 2013.
[17] M. Droppa, B. Matej and M. Haraka, "Cyber threat assessment report in selected environment conducted by choosen technology of firewalls," *Science & Military Journal*, vol. 12, no. 2, pp. 37–41, 2017.
[18] S. S. Chakkaravarthy, D. Sangeetha and V. Vaidehi, "A survey on malware analysis and mitigation techniques," *Computer Science Review*, vol. 32, no. 1, pp. 1–23, 2019.
[19] H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, no. 1, pp. 64366–64374, 2019.
[20] A. M. Hassan, S. Mahmood, M. Alshayeb and M. Niazi, "A maturity model for secure software design: A multivocal study," *IEEE Access*, vol. 8, no. 1, pp. 215758–215776, 2020.
[21] M. A. Hamid, Y. Hafeez, B. Hamid, M. Humayun and N. Jhanjhi, "Towards an effective approach for architectural knowledge management considering global software development," *International Journal of Grid and Utility Computing*, vol. 11, no. 6, pp. 780–791, 2020.
[22] M. Haris, M. Alshayeb, S. Mahmood and M. Niazi, "An empirical study to improve software security through the application of code refactoring," *Information and Software Technology*, vol. 96, no. 1, pp. 112–125, 2018.
[23] H. Sandra, "Are software security issues a result off in software development methodologies," Ph.D. dissertation, Utica College, New York, USA, 2020.
[24] S. Hamza and M. Naveed, "Sok: Anatomy of data breaches," *Proceedings on Privacy Enhancing Technologies*, vol. 4, no. 1, pp. 153–174, 2020.
[25] Positive Technologies, Web Applications vulnerabilities and threats: Statistics for 2019, https://www.ptsecurity.com/ww-en/analytics/web-vulnerabilities-2020/, Accessed on March 2021.
[26] M. Dawson, D. N. Burrell, E. Rahim and S. Brewster, "Integrating software assurance into the software development life cycle (SDLC)," *Journal of Information Systems Technology and Planning*, vol. 3, no. 6, pp. 49–53, 2010.
[27] Y. H. Tung, S. C. Lo, J. F. Shih and H. F. Lin, "An integrated security testing framework for secure software development life cycle," in *Proc APNOMS*, pp. 1–4, Kanazawa, Japan, 2018.

[28] M. Niazi, A. M. Saeed, M. Alshayeb, S. Mahmood and S. Zafar, "A maturity model for secure requirements engineering," *Computers & Security*, vol. 95, no. 1, pp. 101852, 2020.
[29] B. Subedi, A. Alsadoon, P. W. C. Prasad and A. Elchouemi, "Secure paradigm for web application development," in *Proc RoEduNet*, Bucharest, Romania, pp. 1–6, 2016.
[30] M. Alenezi and S. Almuairfi, "Security risks in the software development lifecycle," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 1, pp. 13, 2019.
[31] A. N. Karim, A. Albuolayan, T. Saba and A. Rehman, "The practice of secure software development in SDLC: An investigation through existing model and a case study," *Security and Communication Networks*, vol. 18, no. 9, pp. 5333–5345, 2016.
[32] A. H. A. Kamal, C. C. Y. Yen, G. J. Hui and P. S. Ling, "Risk assessment, threat modeling and security testing in SDLC," arXiv preprint arXiv: 2012.07226, pp. 1–13.
[33] N. Nazir and M. K. Nazir, "A review of security issues in SDLC," *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*, vol. 46, no. 1, pp. 247–259, 2018.
[34] J. Danahy, "The 'phasing-in'of security governance in the SDLC," *Network Security*, vol. 12, no. 1, pp. 15–17, 2008.
[35] M. Nabil, M. Niazi, M. Alshayeb and S. Mahmood, "Exploring software security approaches in software development lifecycle: A systematic mapping study," *Computer Standards & Interfaces*, vol. 50, no. 1, pp. 107–115, 2017.
[36] M. Asadullah, R. K. Yadav and V. Namdeo, "A survey on security issues and challenges in cloud computing," *International Journal of Innovative Research in Technology and Management*, vol. 4, no. 4, pp. 43–50, 2020.
[37] Y. A. Bangash and Y. E. Salhi, "Security issues and challenges in wireless sensor networks: A survey," *IAENG International Journal of Computer Science*, vol. 44, no. 2, pp. 94–108, 2017.
[38] A. Yosef and Q. H. Mahmoud, "Cyber physical systems security: Analysis, challenges and solutions," *Computers & Security*, vol. 68, no. 1, pp. 81–97, 2017.
[39] F. A. Bukhsh, Z. A. Bukhsh and M. Daneva, "A systematic literature review on requirement prioritization techniques and their empirical evaluation," *Computer Standards & Interfaces*, vol. 69, no. 1, pp. 103389, 2020.
[40] A. Ishfaq, M. Asif, M. Shahbaz, A. Khalid and M. Rehman *et al.,* "Text categorization approach for secure design pattern selection using software requirement specification," *IEEE Access*, vol. 6, no. 1, pp. 73928–73939, 2018.
[41] A. Yasemin, C. Stransky, D. Wermke, C. Weir and M. L. Mazurek *et al.,* "Developers need support, too: A survey of security advice for software developers," in *Proc SecDev*, Cambridge, MA, USA, pp. 22–26, 2017.
[42] W. Brandon, S. Mengel and A. Ertas, "An evolutionary approach for the hierarchical scheduling of safety- and security-critical multicore architectures," *Computers*, vol. 9, no. 3, pp. 71, 2020.
[43] A. A. Adanna and O. F. Nonyelum, "Criteria for choosing the right software development life cycle method for the success of software project," *IUP Journal of Information Technology*, vol. 16, no. 2, pp. 39–65, 2020.
[44] M. K. Sharma, "A study of SDLC to develop well engineered software," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 3, pp. 520–523, 2017.
[45] S. U. Rehman and V. Gruhn, "An effective security requirements engineering framework for cyber-physical systems," *Technologies*, vol. 6, no. 3, pp. 65, 2018.
[46] M. Humayun and N. Jhanjhi, "Exploring the relationship between GSD, knowledge management, trust and collaboration," *Journal of Engineering Science and Technology*, vol. 14, no. 2, pp. 820–843, 2019.
[47] S. Pukdesree, "The comparative study of collaborative learning and SDLC model to develop IT group projects," *TEM Journal*, vol. 6, no. 4, pp. 800–809, 2017.
[48] E. I. Tatli, "Developer-oriented web security by integrating secure SDLC into IDEs," *Sakarya University Journal of Computer and Information Sciences*, vol. 1, no. 1, pp. 36–44, 2018.
[49] S. Turpe, "The trouble with security requirements," in *Proc REW*, Lisbon, Portugal, pp. 122–133, 2017.

[50] M. Hamza, H. Hu, M. A. Akbar, F. Mehmood and Y. Hussain *et al.,* "SIOT-RIMM: Towards secure IOT-requirement implementation maturity model," in *Proc EASE*, Trondheim, Norway, pp. 463–468, 2020.
[51] R. Scroggins, "SDLC and development methodologies," *Global Journal of Computer Science and Technology*, vol. 7, no. 7, pp. 1–3, 2014.
[52] S. Rani, "A detailed study of software development life cycle (SDLC) models," *International Journal of Engineering and Computer Science*, vol. 6, no. 7, pp. 22097, 22100, 2017.
[53] E. Aruna, "Design of non-functional requirement (security) using security design patterns in architecture phase to develop secure SDLC," *International Journal of Engineering Research & Technology (IJERT)*, vol. 4, no. 11, pp. 1–5, 2015.
[54] J. C. Santos, K. Tarrit and M. Mirakhorli, "A catalog of security architecture weaknesses," in *Proc ICSAW*, Gothenburg, Sweden, pp. 220–223, 2017.
[55] N. Hernan, J. Antonio and M. Villavicencio., "Systematic mapping of the literature on secure software development," *IEEE Access*, vol. 9, no. 1, pp. 36852–36867, 2021.
[56] V. Mohino, J. B. Higuera, J. R. Higuera and J. A. S. Montalvo, "The application of a new secure software development life cycle (S-SDLC) with agile methodologies," *Electronics*, vol. 8, no. 11, pp. 1218–1246, 2019.
[57] F. Amalia, M. Kurniawan and D. T. Setiawan, "The design of traceability information system of smart packaging-based product supply chain to improve a competitiveness of apple processed agro-industry," *Journal of Information Technology and Computer Science*, vol. 5, no. 3, pp. 247–254, 2020.
[58] A. Imeri and D. Khadraoui, "The security and traceability of shared information in the process of transportation of dangerous goods," in *Proc NTMS*, Paris, France, pp. 1–5, 2018.
[59] X. Xu, Q. Lu, Y. Liu, L. Zhu and H. Yao *et al.,* "Designing blockchain-based applications a case study for imported product traceability," *Future Generation Computer Systems*, vol. 92, no. 1, pp. 399–406, 2019.
[60] K. Cheung and J. Misic, "On virtual private networks security design issues," *Computer Networks*, vol. 38, no. 2, pp. 165–179, 2002.
[61] Y. Ali, Y. Xia, M. Linag and A. Hammad., "Secure design for cloud control system against distributed denial of service attack," *Control Theory and Technology*, vol. 16, no. 1, pp. 14–24, 2018.
[62] A. Z. Henley, K. Muclu, M. Christakis, S. D. Fleming and C. Bird, "Cfar: A tool to increase communication, productivity, and review quality in collaborative code reviews," in *Proc CHI*, Montreal, Canada, pp. 1–13, 2018.
[63] S. Georgiou, S. Rizou and D. Spinellis, "Software development lifecycle for energy efficiency: Techniques and tools," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–33, 2019.
[64] B. C. Zapata, J. L. Aleman, A. Toval and A. Idri, "Reusable software usability specifications for mHealth applications," *Journal of Medical Systems*, vol. 42, no. 3, pp. 1–9, 2018.
[65] B. P. Miller and E. Heymann, "Tutorial: Secure coding practices, automated assessment tools and the SWAMP," in *Proc SecDev*, Cambridge, MA, USA, pp. 124–125, 2018.
[66] B. Hoisl, S. Sobernig and M. Strembeck, "Reusable and generic design decisions for developing UML-based domain-specific languages," *Information and Software Technology*, vol. 92, no. 1, pp. 49–74, 2017.
[67] H. Assal and S. Chiasson, "Think secure from the beginning a survey with software developers," in *Proc CHI*, Glasgow, UK, pp. 1–13, 2019.
[68] A. Hala and S. Chiasson, "Security in the software development lifecycle," in *Proc SOUPS*, Baltimore, USA, pp. 281–296, 2018.
[69] A. Papageorgiou, M. Strigkos, E. Politou, E. Alepis and A. Solanas *et al.,* "Security and privacy analysis of mobile health applications: The alarming state of practice," *IEEE Access*, vol. 6, no. 1, pp. 9390–9403, 2018.
[70] N. Meng, S. Nagy, D. Yao, W. Zhuang and A. A. Gustawo, "Secure coding practices in java: Challenges and vulnerabilities," in *Proc ICSE*, Gothenburg, Sweden, pp. 372–383, 2018.
[71] D. R. Lakshmi and S. S. Mallika, "A review on web application testing and its current research directions," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 4, pp. 2132, 2017.
[72] K. Sneha and G. M. Malle, "Research on software testing techniques and software automation testing tools," in *Proc ICECDS*, Chennai, Tamil Nadu, India, pp. 77–81, 2017.

[73] J. Choliz, J. Vilas and J. Moreira, "Independent security testing on agile software development: A case study in a software company," in *Proc ARES*, Toulouse, France, pp. 522–531, 2015.
[74] T. Rangnau, R. Buijtenen, F. Fransen and F. Turkmen, "Continuous security testing: A case study on integrating dynamic security testing tools in CI/CD pipelines," in *Proc EDOC*, Eindhoven, Netherlands, pp. 145–154, 2020.
[75] P. P. Kumar, "Development of software testing techniques for early fault detection," *Journal of Advancement in Parallel Computing*, vol. 3, no. 3, pp. 15–23, 2021.
[76] A. K. Alvi, M. Zulkernine, "A security pattern detection framework for building more secure software," *Journal of Systems and Software*, vol. 171, no. 1, pp. 110838–110860, 2021.
[77] S. Malik, "Software testing: Essential phase of SDLC and a comparative study of software testing techniques," *International Journal of System & Software Engineering*, vol. 5, no. 2, pp. 38–45, 2017.
[78] P. Dehraj, A. Sharma and P. Grover, "Maintenance assessment guidelines for autonomic system using ANP approach," *Journal of Statistics and Management Systems*, vol. 22, no. 2, pp. 289–300, 2019.
[79] A. Wang, H. Wang, B. Jiang and W. K. Chan, "Artemis: An improved smart contract verification tool for vulnerability detection," in *Proc DSA*, Xi'an, China, pp. 173–181, 2020.
[80] S. M. Rahaman and A. Kumari, "A model for corrective software maintenance effort estimation after privacy leak detection in social network," in *Proc AISP*, Amaravati, India, pp. 1–10, 2020.
[81] M. Bellare, J. A. GAray, R. Hauser, A. Herzberg and H. krawczyk *et al.,* "Design, implementation, and deployment of the iKP secure electronic payment system," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 611–627, 2000.
[82] P. Frijns, R. Bierwolf and T. Zijderhand, "Reframing security in contemporary software development life cycle," in *Proc ICTMOD*, Marrakech, Morocco, pp. 230–236, 2018.
[83] R. A. .Khan, S. U. Khan, H. U. Khan and M. Ilyas, "Systematic mapping study on security approaches in secure software engineering," *IEEE Access*, vol. 9, pp. 19139–19160, 2021.
[84] J. P. Donnelly, "A systematic review of concept mapping dissertations," *Evaluation and Program Planning*, vol. 60, no. 1, pp. 186–193, 2017.
[85] A. J. Canas, P. Reiska and A. Mollits, "Developing higher-order thinking skills with concept mapping: A case of pedagogic frailty," *Knowledge Management & E-Learning: An International Journal*, vol. 9, no. 3, pp. 348–365, 2017.
[86] R. D. Mauricio, L. Veado, R. T. Moreira, E. Figueiredo and H. Costa, "A systematic mapping study on game-related methods for software engineering education," *Information and Software Technology*, vol. 95, no. 1, pp. 201–218, 2018.