IJEMD-CSAI, 2 (1) (2023), 1 – 18 https://doi.org/10.54938/ijemdcsai.2023.02.1.256

International Journal of Emerging Multidisciplinaries:
Computer Science and Artificial Intelligence

Research Paper
Journal Homepage: www.ojs.ijemd.com
ISSN (print): 2791-0164 ISSN (online): 2957-5036

Secure Software Development Best Practices

Muhammad Firdaus Fauzi[1], Vinod Rama Mohan[1], Yang Qi[1], Christal Chandrasegar[1], Saira Muzafar[1*]

[1] School of Computer Science, Taylor's University, Subang Jaya, Selangor, Malaysia
**Corresponding author*

Abstract

This research aims to explore optimal strategies for fortified software, enhancing the implementation of secure software development practices. Software security involves crafting and designing software that guarantees the integrity, confidentiality, and availability of its code, data, and functionalities. Often, in prioritizing functionality, security takes a back seat when organizations embark on system development. Yet, it's imperative to embed security at every phase of the Software Development Life Cycle (SDLC). Numerous methodologies and models exist for addressing software security, but only a few substantiate creating secure software applications effectively. Despite advancements, software security remains inadequately addressed, posing a challenge to integrating security protocols into the SDLC seamlessly. This review advocates specific security measures to be integrated at each SDLC level, fostering a secure SDLC. Efficient amalgamation of these processes ensures the delivery of secure software systems with minimized resource expenditure. Additionally, it highlights hurdles encountered in employing agile development methodologies for crafting secure software. These challenges stem from assessing agile ideals, principles, and security assurance procedures. These findings underscore the urgency for research facilitating safe software development, addressing barriers inhibiting its adoption. The paper serves as a valuable reference, shedding light on the significance of establishing secure software development processes.

Keywords: SDLC; Secure Software Development; Secure Software Development Life Cycle; Software Security.

Email addresses: muhammadfirdausfauzi@sd.taylors.edu.my (M.F.Fauzi), vinod.ramamohan@sd.taylors.edu.my (V.R.Mohan), yangqi@sd.taylors.edu.my (Y.Qi), christalanne.chandrasegar@sd.taylors.edu.my (C.Chandrasegar), sairamuzafar@sd.taylors.edu.my (S.Muzafar)

1. Introduction

Secure software is one of the most crucial aspects in any software development or service and the protection the IT infrastructures[1], it is the stepping stone for a quality product or service and may save considerably a significant amount of funds in the long run when developing a software product or service [2][3]. To elaborate, software development security is similarly important as compared to the functionalities of a software. Software security ensures that the CIA (Confidentiality, Integrity, Availability) of data and services are not compromised. Security is a crucial aspect that necessitates careful consideration right from the inception of the software development process. Bugs and faults identified in the early stages of development are more easily and cost-effectively addressed in comparison to those discovered at later Hence, the integration of security measures throughout the whole Software Development Life Cycle (SDLC) is necessary in order to provide secure conventional practices, security is commonly regarded as a secondary concern that is addressed by developing patches for any vulnerabilities discovered during project testing or post-deployment [4] . Briefly, software security may be defined as the protection of software applications from unauthorized access, use, or destruction. Often compared to cybersecurity, software security techniques are applied during software development unlike cybersecurity which focuses on protecting internet-based systems. In focus, the goal of software development security is to ensure that the application is secured and remains functional under a malicious attack [5]. Utilizing a safe software product in an actual setting allows the system's overall susceptibility to be decreased. It would make sense to determine the best way to incorporate security practices such as secure code into the traditional SDLC [6]. In this section we provide a detailed introduction to the current secure software development trends which includes the development process and the tools/strategies used.

The core principles of software security describe factors such as methods, frameworks, and strategies implemented to defend a software against attacks by reducing vulnerabilities that attackers may exploit on [7]. There are methods that developers use as guidelines when developing a secured software which is known as the *Software Development Life Cycle (SDLC)*. In detail, the SDLC method includes phases which are planning, requirement gathering and analysis, designing, coding, testing, deployment, and maintenance.

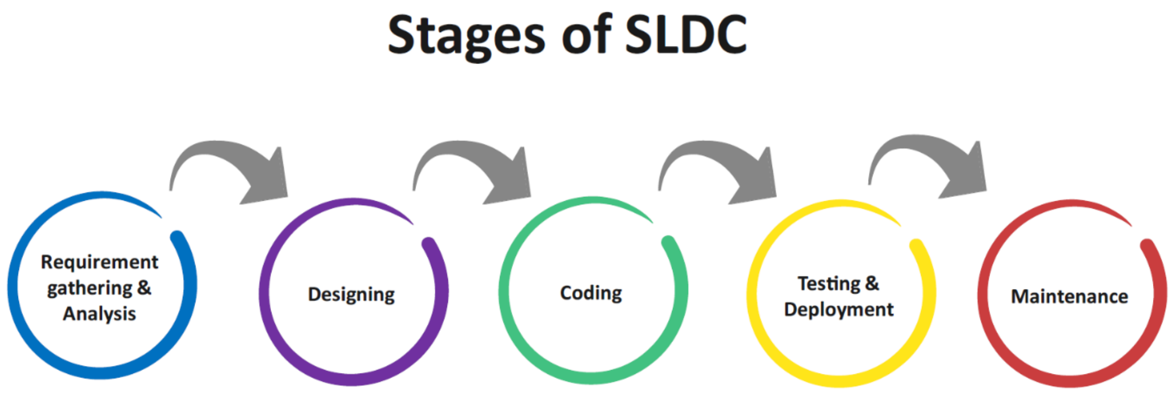

Figure 1: Stages of SDLC

1.1. SDLC Models

Among the different types of SDLC models that are mainly used are Waterfall model, V-shaped model and Agile SDLC model [8].

a. Waterfall

Firstly, the Waterfall model or linear sequential model is an early model that is used in SDLC. The waterfall model is better suited for general systems or software that takes the shape of software that can offer services to the customer [9]. In this model, the outcome of a phase is the input for the following phase where development of the next phase can only begin when the respective previous phase has been completed. The waterfall model consists of 6 phases which are requirement gathering and analysis, system design, implementation, integration and testing, development of system and maintenance. An advantage is that it is easy to understand as the phases are progressed step by step whereas a disadvantage would be that this model is time-consuming due to having each phase to be completed prior to moving on to the next. Each of these phases have their respective unique objectives which are elaborated as below [10].

1. **Requirement gathering and analysis:** focuses on the brainstorming, gathering and documentation of possible requirements of the system.
2. **System design**: The overall system architecture is defined by focusing on the preparation of the system design resulting in aiding the hardware and system requirements.
3. **Implementation:** The system is developed or built in discrete programmes called units, with input from the system design, and then integrated in the following phase. Each unit is developed and tested based on its functionality which is also known as unit testing.
4. **Integration and testing:** After testing each unit, all the units developed during the implementation phase are merged into a system. The system is then tested for failures or errors post integration.
5. **Deployment of system:** After the completion of testing the functional and non-functional properties, the product is deployed in the customer environment or launched to the market.
6. **Maintenance:** In the case where a few issues that arise in the client environment. Patches are released to address these vulnerabilities. To improve the product, newer and updated versions are issued. Improvements to the client environment are brought upon maintenance of the system.

b. V-shaped model

The following SDLC model is the V-shaped model, which is also known as the verification and validation model. Whereby, development and testing phases are executed in parallel. Additionally, V-model and Waterfall model are similar except that the test planning and testing start an early stage in V-model [11]. It can be categorized into two groups which are the verification phase and validation phase. **Table 1.1** shows the description of each stage in the verification and validation phase.

Table 1.1: V-model specifications [12]

Verification		Validation	
Stage	**Description**	**Stage**	**Description**
Requirement analysis	Required information is gathered and analysed.	**Unit testing**	Performed on individual components. Detects early defection. Managed in low-level design phase.
System design	Architecture, components, design are created and documented.	**Integration testing**	Executed the design phase (high level) Testing activities is done on integrated modules.
High-level design	The design of each respective modules and its integrated functionality between them.	**System testing**	Performed during the system design phase. Entire system functionality is tested.
Low-level design	Design and architecture of each component.	**Acceptance testing**	Related to the requirement analysis phase. Performed in the customer's environment.
Coding	Code development		

c. Agile Method

Agile SDLC is a prominent development approach that focuses on process flexibility and customer satisfaction by delivering a functioning software product to the customers quickly. This enables the customers to understand how the software works after it has been completed and to provide any requirements or suggestions for modifying the application, even when it is still in the late development stage. The Agile SDLC process flow is divided into five stages*: concept, inception, iteration, release, and production*. During the concept phase, the project is brainstormed and prioritised. After careful study, the process moves on to inception, during which teams are created, the fundamental environment is negotiated, and a budget is produced. During iteration, the team of developers works extensively to consistently provide functional products to customers and make modifications based on their requests and feedback. The release phase follows, during which quality assurance testing is undertaken to obtain the final version iteration of the product. Finally, the production phase is the period in which the system is operated and maintained. Agile SDLC is favourable in terms of flexibility since the entire project is divided into tiny iterations that may make changes often. As a result, development risk is reduced, and customer satisfaction is ensured because they may provide ongoing input to the software [13].

2. Literature Review

In this section, numerous primary study sources were reviewed and analysed in relation to the current secure software issues and challenges. This section provides an in-depth analysis of multiple research articles to conclude and identify outcomes of their respective research papers to obtain an understanding on the challenges and issues faced when developing a secured software These study sources are utilized to gain an understanding of the existing research while challenging their arguments and debates to challenges faced in developing a secured software and its best practices.

2.1. Key takeaways of study sources

According to the research conducted by Rafiq Ahmad Khan et. al., security development at each phase of the Software Development Life Cycle (SDLC) is crucial and becoming an urgent requirement[14]. To add, several efforts such as methodologies, strategies, and models have been suggested to address these issues but only a few are deemed reliable. The purpose of their research is to analyse about software security risks and practices to promote better designed secure software development methods. In this study, several prescribed security activities in each SDLC phase were discussed to pursue a secure SDLC. It was concluded from that post-development phases in securing software systems were insufficient. To summary, software security is a critical aspect that should be prioritised. Due to the poor focus on security, many software projects have failed in the past. Testing software security can be complex and expensive especially if it is done after it has been produced and launched [15]. Secure software engineering feels that software security is an important aspect that should be evaluated early in the SDLC process. The authors emphasized that security features are to be integrated into an application development life cycle while adapting the latest secured software engineering practices [16].

Another study published by Hela Queslati et. al, it was found that multiple challenges arose when developing a secure software with the agile development approach. These issues were identified, and the causes were evaluated based on the agile values, principles, and security assurance practices [17]. The authors mentioned 20 challenges where 14 of these were related while 6 of them were not caused by the agile values, principles nor by the security assurance practices. It was found from their research that the agile development approach is only using a short amount of time to produce shippable working software iterations. It was reported that the agile development methods also frequently have requirement changes with more deliveries while developing secure software needs the use of verification gates and refinement of artefacts. One of the few challenges reported in this paper was that of the lack of integration of security requirements was present due to the fast-paced shipping of iterations resulting in security measurements to not be implemented properly due to time constraints [18], [19].

Furthermore, the study conducted by Simon Foley et. al proposed that one of the main challenges faced in developing a secure software is regarding the developers itself. It was reported that their research is based on the developer-centred security issues and the challenges faced where developers are required to consider the security measures from the perspective of other developers who utilize their software. The main discussion of their paper focused on whether current user-centred security techniques are sufficient where two directions were suggested to consider this challenge which are symmetry of ignorance and security archaeology [20]. One of the main discussions of this research was that software developed by individual developers adopted a more centric approach, as a result, arising the issue of developers not being omniscient or all-knowing in areas of software developing and hence tends to lack the necessary security requirements for a secure software. While an expert in their field, individuals may cause or face security vulnerabilities by ignoring or being unaware of some detail of an artefact that they employ to solve their issues. Furthermore, they are not confident on how to ensure that the callers of their code would not repeat the same action. The problem is to guarantee that the developer of any product provides it with all appropriate and sufficient security, from the perspective of people who utilise the product while being unaware of its underlying intricacy [21].

Based on the paper published by Zulfikar Ahmed Maher et. al., it was found that security measurements were not consistently addressed in the initial design phase of the software development. Hence, resulting in an abundance of software systems with penetrable security defences [22]. The research revolved around studying the factors that influenced the developer's intention to adopt such secure software development practices. The methods practiced for their research was based on qualitative research methodology where interviews were conducted to gain an understanding from professionals working at Malaysian software development organizations. The findings from their interviewees and report resulted in several challenges. Whereby, the first obstacle to implementing and practicing secure software development, is the lack of vision and clear directions supplied by the senior management, with a low focus on the security principles required in designing systems. Another important concern raised was the absence of security project implementation. As with the implementation of the agile technique, this research noted the issue of short deadlines because of a lack of attention on security measures [23].

It is noted from literature review, that authors have similar yet slightly different views on the issues and challenges faced when developing a secure software. The four main issues derived and concluded are firstly, Rafiq Ahmad Khan et. al found that software developments lack security measures in the early stages of the SDLC process and post-SDLC process due to ignorance of developers. Secondly, the study by Hela Queslati et. al proposed that the challenges faced were due to the inefficiency of the agile SDLC approach of fast-shipping iterations which caused security measurements to be implemented without quality due to time constraints as one of the main challenges. Thirdly, the study conducted by Simon Foley et. al found that the issues faced when developing secure software were mainly due to the ignorance and centric approach of developers which tend to increase security vulnerabilities in their software due to lack of knowledge in other areas of security development. Finally, the study conducted by Zulfikar Ahmed Maher et. al. suggested that the cause of issues in secure software development were due to the lack of vision and clear directions by senior management and the time constraints to implement security requirements into a software.

2.2. The challenges of lack of security integration in SDLC phases

As previously discussed, the research conducted by Rafiq Ahmad Khan et. al. focuses on the issue of lack of security integration in the SDLC phases. This section elaborates on the security issue at each part of the SDLC phase that was found in the report. The identified security risks, along with the frequencies of each risk are discussed in the following subsections.

a) Requirement Engineering (RE) phase

It was reported that the security risks in the RE phase of the SDLC model rated highly as a factor of the development of secure software. Specifically, it was reported as the top amongst all the identify security risks at approximately 97.5% [14]. Certain security risks might occur if security is not integrated from the beginning of the SDLC phase, as obtained from their report and several literatures, this is listed in **Table 2.1** as shown below which shows the top 10 types of security risks in the RE phase from the 25 types. The main contributor of security issues in this phase is due to the ignorance of security leading to poor security management.

Table 2.1: Security risks in the Requirement phase [14]

No.	Security Risks	Frequency
1	Ignored security requirements	91 sources
2	Improper security requirements negotiation and management	56 sources
3	Poor security requirements validation	49 sources
4	Neglected risk assessment	34 sources
5	Improper security risk analysis	33 sources
6	Poor security requirement documentation	26 sources
7	Neglected threat modelling development	25 sources
8	Lack of security requirements data collection activity	21 sources
9	Improper security requirement analysis and inception	20 sources
10	Poor secure requirement documentation	15 sources

b) Design phase

Mostly found the software bugs and issues are present in the design phase of the SDLC process [24]. The design process in the SDLC model is the foundation for developing a secure software system [25]. Hence, by reducing risks in this phase, significant effort needed maybe reduced in the following phases. As shown in **Table 2.2**, the top 5 most common security issues that arise in the design phase are listed with its respective frequencies.

Table 2.2: Security risks in the design phase [6]

No.	Security Risks	Frequency
1	Neglected development of threat modelling	57 sources
2	Poor attention of following security design principles	29 sources
3	Neglected security activities such as awareness and training	27 sources
4	Lack of documentation of secure design	23 sources
5	Poor integration of utilizing attack patterns and understanding abuse case models	23 sources

c) Development/Coding phase

Poor secure development or coding remains as one of the prominent challenges in the SDLC process. This is because that it may be challenging to select a proper programming language and classification of modules to be used. A more secure code being released is based on the inclusion of security protections in each phase of the SDLC. Poor integration of authentication and authorization modules are due to the poor implementation of authentication functions and access-control policies [26]. Further, authentication and authorizations are crucial components of a basic secure software system. Threat modelling at Microsoft is often based on the *STRIDE* model which account for *Spoofing (S), Tampering (T), Repudiation (R), Information disclosure (I), Denial-of-service (D), and Elevation of privilege (E)* [27]. Some of the most prevalent security vulnerabilities that impede the process of secure authorization and authentication include tampering, spoofing, repudiation, information exposure, denial-of-services[28] [29], privilege elevation, and failure to restrict URL access or access control. **Table 2.3** shows the top 10 security risks in the development phase.

Table 2.3: Security risks in the development phase [6]

No.	Security Risks	Frequency
1	Data tampering	54 sources
2	SQL Injection	37 sources
3	Cross Site Scripting or Cross-site request forgery (XSS)	35 sources
4	Denial-of-service (DoS)	32 sources
5	Repudiation	29 sources
6	Information disclosure	29 sources
7	Elevation of privilege	28 sources
8	Spoofing	26 sources
9	Password conjecture: poor password complexity management	26 sources
10	Buffer and array overflow	25 sources

c) Security testing phase

In the testing phase of the SDLC, the functionalities of each system components are ensured that they provide it individually and integrated to the entire system. It was also reported that the SDLC phase is the most time-consuming and costly process of the SDLC due to its complexity [30]. **Table 2.4** lists some of the common security risks in the security phase.

Table 2.4: Security risks in the security phase [6]

No.	Security Risks	Frequency
1	Improper penetration testing analysis	32 sources
2	Poor dynamic and static security testing analysis	30 sources
3	Improper review of final security design	26 sources
4	Fuzz testing not implemented	16 sources
5	Brute force attack testing not implemented	7 sources
6	Threat modelling executed improperly	7 sources
7	Neglected function/non-functional testing	6 sources
8	Neglected manual code reviewing	6 sources

d) Deployment phase

A few of the challenging issues in the developing phase are designing authentication protocols, improper configuration management, developing strong cryptosystems, building effective trust models and the policies in the security [31]. According to Rafiq Ahmad Khan et. al, this phase reported the least of issues compared to the other phases in the SDLC model. Hence, **Table 2.5** shows some of the common software security risks from the deployment phase.

Table 2.5: Security risks in the deployment phase [6]

No.	Security Risks	Frequency
1	Poor default software configuration	9 sources
2	Improper implementation of logout	5 sources
3	Enabled services and ports were setup improperly	3 sources
4	Security failures are neglected	3 sources
5	Validation on output executed poorly	3 sources
6	Secure APIs are unused or neglected	2 sources
7	Poor security feedback review	2 sources
8	Planning and execution response were neglected	2 sources

e) Maintenance phase

A wide variety of possible vulnerabilities may be evaluated through vulnerability-oriented architectural research which can provide a systematic and thorough approach. However, this approach was reported to be time-consuming and costly, especially this can be an issue where software development iterations are time constrained which makes fitting security activities challenging due to the time-consuming nature of it. Some of the common problems that arise due to time pressure are the compromise of security integration to accelerate the launching schedule, timing attacks, insufficient time for teams to adapt to the security activities and tight deadlines which promotes pressure. Hence, **Table 2.6** shows the reported issues concerning security risks in the maintenance phase.

Table 2.6: Security risks in the maintenance phase [6]

No.	Security Risks	Frequency
1	Poor security trust	18 sources
2	Improper methods use to discover new threats	15 sources
3	Improper utilization to discover attack surface area for new threats	13 sources
4	Neglecting security patches for new threats	12 sources
5	Poor change control, vulnerability management and configuration	8 sources
6	Increased software cost due to security activities	7 sources
7	Excessive and threatening timing attacks	5 sources
8	Software updates and username/password changes could not be made	5 sources
9	Lack of log analysis and optimization	4 sources
10	Poor user knowledge due to lack of education in practicing software application securely	4 sources

2.3. The challenges of using the Agile process

The challenges arise using agile software development method can be categorised into five groups namely *Software development life-cycle challenges*, *Incremental challenges*, *security assurance challenges*, *Awareness collaboration challenges and Security management challenges* [17]. This section elaborates and discusses each of these challenges, it is important to mention that these challenges are caused by either agile value or agile principle.

a) Software Development Life Cycle challenges

As previously mentioned, Agile Software Development (ASD) methods focus on developing software in successive iterations [32], however, security requirements elicitation activities were not integrated, nor risk assessment activities were considered in these methods [33] [34] [35]. Additionally, it was also claimed that certain security elicitation activities are required to be repeated for each iteration due to each iteration having to include the full development life cycle. It was mentioned that development iterations are infamous for its tight deadlines which make fitting security activities to be a challenge.

b) Incremental challenges

The security requirements and functional requirements of a software are often inversely related to each other, hence, making security requirements a constraint to the functional requirements of a software [36]. Additionally, it was claimed that the code refactoring that agile methods utilize may disconnect such constraints where they also claimed that continuous code changes may limit the tendency to finish security assurance activities [37]. This argument was also supported by Wayrynen et. al. whereby iterating the changing requirements and design may break the system security requirements [33]. Hence, the presence of these changes make tracing the requirements to the security objectives challenging, an opinion according to Bartsch [19].

c) Security assurance challenges

ASD methods support the use of light documentations and hence resulting in conflicts with the use of documentation for security assessment, claimed by Beznosov and Kruchten [37], Hoeren [38], and Alnatheer et. al. [39]. Also agile developers' test methodology (relying on tests to check that requirements are fulfilled) contradicts with security demands since tests are, in general, insufficient to assure the application of security requirements [37]. Additionally, it is also hard to cover all vulnerability cases while it also being hard to automate. In an agile approach, security audits may be challenging due to the practice of improving the development processes when considering the lessons learned. For example, certain activities used for the auditing process may be discontinued and the used resources may become not uniform [19].

d) Collaboration awareness challenges

Developer and customer collaboration is encouraged in the ASD approach which requires all project members to be educated about the foundation of developing a secure software. Unfortunately, in this case, agile developers require further training about developing secure software [19] where security practices are often neglected in favour of functional requirements. This argument was supported by Wayrynen et al [33], Hoeren [38], and Alnatheer et al. [39].

e) Security management challenges

The relationship between a security in a software and its products’ cost are directly proportional due to an increased development effort [39]. Additionally, organizations also neglect security activities to accelerate the release scheduling and to support this, it was reported that no incentives were given for organizations to develop security features in the early increments, claimed by Bartsch [19] and Hoeren et al. [38].

f) Summary table of challenges using Agile approach

Table 2.7 shows the summary of the previously discussed challenges using the Agile approach.

Table 2.7: Summary of causes of challenges using Agile approach [17]

Category Challenges	**Challenge**
SDLC	*Security activities are to be implemented for each development iteration.*
	Limited iteration time which may not fit security activity time duration.
Incremental development	*Security constraints are broken due to refactoring.*
	System security requirements may break due to frequent system or software requirement changes.
	Frequent code changes make completing the assurance activities challenging.
	Tracing the requirements to the security objectives are difficult due to requirement changes.
Security assurance	*Security assessment does not favour light documentation*
	Generally, tests are insufficient to ensure integration of security requirements.
	All vulnerability cases may not be covered in tests
	Frequent changing of development processes conflicts with audit needs of uniform stable processes.
Awareness and collaboration	*Neglected security requirements.*
	Objective results need the developer role and security reviewer to be separated.
Security management	*Software costs are increased due to security activities.*
	Security activities are compromised in favour of earlier releasing.

3. Results and Discussion

Based on the literature review conducted, it was found and concluded that there were mainly two major issues that arise when developing a secure software. Whereby, these issues were deemed as lack of security integration in the SDLC phases and the challenges of using the agile approach when developing a secure software. Some researchers mention issues regarding developer-centred security approach and the issue with organizations favouring software functionality as compared to security in a software. However, these factors were not discussed in detail and only mentioned in the *Key takeaway section*. It was also found that most of the challenges faced were during the initial phases of the SDLC model, the development/coding phase, and the post-development phase. It was cited that most developers are ignorant towards security measures and take this topic too lightly, hence, resulting in improper security integration towards developing a software as mentioned in the agile approach. This section of the report aims to provide a discussion based on the findings of the current studied sources by relating it to real world challenges and the software development industry.

Firstly, the lack of integration of security measures in the SDLC has been concluded as one of the main causes of concern when developing a secure software. From the study conducted, poor attention was given to the security aspect of software development and organizations would also cut security activities to save cost and time in favour of software functionalities. As mentioned, the security risks were analysed in each stage of the SDLC model which includes the *Requirement, Design, Development, Testing, Deployment and Maintenance phase.* From our study, the highest reported security issue for each phase were Neglected security requirements, neglected threat modelling, Data tampering, Neglected penetration analysis, improper default software configuration and lack of security trust for the requirement, design, development, testing, deployment, and maintenance phase respectively. The complete table which provides the top reported security threats for each phase can be found under section *2.0 Literature Review*. The lack of attention of security for these phases may pose a huge threat towards the overall protection of the software. Without proper security measures, software is prone to attacks, bugs, and exploits. In fact, mitigating found threats in a software are more costly compared to conducting security activities in the early stages of the SDLC [40].

Furthermore, another factor that is prone to cause complications in secure software development is the poor integration of the agile approach technique. The software industry faces several challenges in developing a secure software which results in organizations to use the Agile Approach to develop their secure software to accelerate their releasing schedule to meet consumer needs. However, most organizations that utilize the agile approach often neglect security activities in favour of software functionality and to release their product earlier. Alternatively, organizations often want to obtain a cheaper and faster product release and development at the cost of security in a software [36]. Developers also tend to emphasise the functional requirements compared to the security requirements in the implementation phase. While being unskilled or weak in the software security field, thus adding security experts to their project teams will be advantageous. The absence of integration of security requirements through fast-paced iterative delivery, which made security activities challenging owing to limited iteration time, may result in the developed secure software having insufficient security quality [41].

4. Proposed Solutions

The proposed solution is based on using the agile approach from secured software development where it has posed many challenges thus to give a proper solution for it, it would be much easier to follow the methods. When developing using the software and using the agile method, it is much more different approach to the overall process in reasons of how certain backlogs are being built. Although, it can help in the development for a better purpose as well. With many complications being imposed by the criteria; it is better to go for solutions so that it can overcome the future and current problems that can be faced. The first part is by implementing a security-related policy where the policy acts as a secure rule or regulation that is needed to be follow when conducting a development using the two approaches. With this being said, when implementing this policy, it should be security-related assurance based as well as a policy that is a benefactor to both developers and the process of it. When the policies are implemented, the best alternatives are chosen as well as best methods used to gain and reduce certain unwanted issues. This would help in cutting costs as well as lessen the workload when working on certain projects based on the approach. This also will help in securing the data or any vital information that is being used. Next, empowering the developers to make sure they are responsible for the security's application. This is where the developers are to be careful when developing applications or software using vital data. In cases of security breaches or data stolen, the developers are to take accounted for the loss as well. This means when conducting such development, the developers are in need of using the best practices of security to protect the data in the process. This method is to be done when the developers can understand the ways and methods to protect and make sure the data is well protected in the process. Thus, it would help in making the application more secure as well, as the data within the application. Furthermore, the proposed solution is to make sure it is more flexible in building a culture of security for the many applications that are being developed in the near future. When implementing security, it becomes more secure, and more applications are better from the users preceptive. This being said that security plays an important role in many applications especially in this era that is full of technology-based products. With this, having security layer helps keeps the mind at ease as well as knowing that the user's data is protected.

Following that, best practices can also be implemented during the different stages of the SDLC model namely the *requirement, designing, implementing, testing, deploying and maintenance phase.* For a secure requirement phase, the purpose is to provide a complete security by integrating basic security functions that include confidentiality, integrity, and availability [19]. The next phase to be discussed is the designing phase, which is one of the creative stages of the SDLC. Almost half of the software defects are identified and detected during the design phase. Briefly, some of the common best practices in the design phase are developing threat models, secure design documentations, understanding and following security design principles, and lastly secure design review and verification [42-48]. For the implementation phase, 80 percent of system penetration is due to coding errors in commercial software [38] [56-60]. Best practices for this phase incudes performing code reviews, provide security knowledge and training to software developers, implement static code analysis and conduct source code assessment process [43] [49-55]. Briefly, the best practices that can be made for the testing, deploying, and maintenance phase are penetration testing, static analysis, and threat database preparation respectively.

5. Conclusion

This paper first introduces the three different types of SDLC models, then analyses the problems and challenges, follows findings and results, and finally lists the proposed solutions. The three popular SDLC models mainly used are Waterfall model, V-shaped model, and Agile SDLC model. Although these three models have distinct differences in structure, strengths & weaknesses, and attention to focus, but they can eventually achieve the same effect of helping developers minimize security vulnerabilities and protect their software from attacks when developing software. The research showed the importance of SDLC for secure softening development and the need of increasing the security activities in each SDLC phase to pursue a better SDLC rule. The main challenge in developing secure software is about the developers themselves, and the impact of the human factor on secure software development is very strong. The research implications also shows that software security measurements are not implemented with the appropriate level of attention at the initial stage today, and the occurrence and consequences of this phenomenon is confirmed by interviewing some software developers in Malaysia. We conclude that the two main problems faced in developing secure software are the lack of integration of security measures in the SDLC and the poor integration of agile methods and techniques, which can easily lead to complications in secure software development. The first problem is that not enough attention is paid to the security aspects of software development and organizations cut security activities to save cost and time in favour of software functionality. The second problem is that organizations want to release software faster / cheaper and meet the needs of consumers, so they choose agile SDLC methods to develop their software and ignore most of the security activities to achieve the above results. In order to avoid these situations happening, the solutions we can offer are to ensure that the software development team follows the secure SDLC rules by issuing relevant security development policies, and then making the relevant developers concerned responsible for the various losses caused by security breaches, therefore forcing them to complete the entire security measurement process.

References

[1] Muzafar, S. & Jhanjhi, N.Z. "Success Stories of ICT Implementation in Saudi Arabia," *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-7998-1851-9.ch008,* 151–163, Jan. 1AD, doi: 10.4018/978-1-7998-1851-9.CH008 (2020).

[2] Humayun, M., Niazi, M., Jhanjhi, N., Alshayeb, M. & Mahmood, S. "Cyber Security Threats and Vulnerabilities: A Systematic Mapping Study," *Arab. J. Sci. Eng.*, **45**(4), 3171–3189, doi: 10.1007/s13369-019-04319-2. (2020).

[3] Muzafar, S., Humayun, M. & Hussain, S.J. "Emerging Cybersecurity Threats in the Eye of E-Governance in the Current Era," *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-7998-9624-1.ch003,* 43–60, Jan. 1AD, doi: 10.4018/978-1-7998-9624-1.CH003 (2022).

[4] M. Humayun, M. Niazi, N. Z. Jhanjhi, S. Mahmood, and M. Alshayeb, "Toward a readiness model for secure software coding," *Softw. Pract. Exp.*, **53**(4), 1013–1035, doi: 10.1002/SPE.3175 (2023).

[5] "What is software security and why is it important? | Contentful."

https://www.contentful.com/blog/software-security-to-deliver-digital-experiences-fast/ (accessed Oct. 04, 2023).

[6] Jakimoski, K., Stefanovska, Z. & Stefanovski, V. "Optimization of Secure Coding Practices in SDLC as Part of Cybersecurity Framework," *J. Comput. Sci. Res.*, **4**(2), 31–41, doi: 10.30564/JCSR.V4I2.4048 (2022).

[7] Saeed, S., Jhanjhi, N.Z., Naqvi, S.M.R. & Khan, A. "Cost Optimization of Software Quality Assurance," *Stud. Big Data*, **91,** 241–255, doi: 10.1007/978-3-030-75855-4_14/COVER (2022).

[8] "What Is SDLC (Software Development Life Cycle) Phases & Process." https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/ (accessed Oct. 04, 2023).

[9] Rachma, N. & Muhlas, I. "Comparison Of Waterfall And Prototyping Models In Research And Development (R&D) Methods For Android-Based Learning Application Design," *J. Inov. Inov. Teknol. Inf. dan Inform.*, **5**(1), 36–39, doi: 10.32832/INOVA-TIF.V5I1.7927 (2022).

[10] "SDLC - Waterfall Model." https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model (accessed Oct. 04, 2023).

[11] Saeed, S., Jhanjhi, N.Z., Naqvi, M. & Humayun, M. "Analysis of software development methodologies," *Int. J. Comput. Digit. Syst.*, **8**(5), 445–460, doi: 10.12785/IJCDS/080502 (2019).

[12] "SDLC - V-Model." https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm (accessed Oct. 04, 2023).

[13] "SDLC - Agile Model." https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm (accessed Oct. 04, 2023).

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

[15] Bracciale, L., Loreti, P., Detti, A., Paolillo, R. & Melazzi, N.B. "Lightweight named object: An ICN-based abstraction for IoT device programming and management," *IEEE Internet Things J.*, **6**(3), 5029–5039, doi: 10.1109/JIOT.2019.2894969 (2019).

[16] Zhang, M., X., Carnavalet, X.C., Wang, L. & Ragab, A. "Large-scale empirical study of important features indicative of discovered vulnerabilities to assess application security," *IEEE Trans. Inf. Forensics Secur.*, 14(9), 2315–2330, doi: 10.1109/TIFS.2019.2895963 (2019).

[17] Oueslati, H., Rahman, M.M. & Ben Othmane, L. "Literature Review of the Challenges of Developing Secure Software Using the Agile Approach," *2015 10th Int. Conf. Availability, Reliab. Secur.*, 540–547, doi: 10.1109/ARES.2015.69 (2015).

[18] Keramati, H. & Mirian-Hosseinabadi, S.H. "Integrating software development security activities with agile methodologies," *AICCSA 08 - 6th IEEE/ACS Int. Conf. Comput. Syst. Appl.*, 749–754, doi: 10.1109/AICCSA.2008.4493611 (2008).

[19] Bartsch, S. "Practitioners' perspectives on security in agile development," *Proc. 2011 6th Int. Conf. Availability, Reliab. Secur. ARES 2011*, 479–484, doi: 10.1109/ARES.2011.82 (2011).

[20] Pieczul, O., Foley, S. & Zurko, M.E. "Developer-centered security and the symmetry of

ignorance," *ACM Int. Conf. Proceeding Ser.*, 46–56, doi: 10.1145/3171533.3171539 (2017).

[21] N. Cross, "Developments in design methodology," 357, 1984.

[22] Shah, A. et al. "Challenges and limitations in secure software development adoption - A qualitative analysis in Malaysian software industry prospect," *Indian J. Sci. Technol.*, **13**(26), 2601–2608, doi: 10.17485/IJST/V13I26.848 (2020).

[23] Geer, D. "Are companies actually using secure development life cycles?," *Computer (Long. Beach. Calif).*, 43(6), 12–16, doi: 10.1109/MC.2010.159 (2010).

[24] Maheshwari, V. & Prasanna, M. "Integrating risk assessment and threat modeling within SDLC process," *Proc. Int. Conf. Inven. Comput. Technol. ICICT 2016*, **1,** doi: 10.1109/INVENTIVE.2016.7823275 (2016).

[25] Khan, R.A., Khan, S.U., Khan, H.U. & Ilyas, M. "Systematic Mapping Study on Security Approaches in Secure Software Engineering," *IEEE Access*, **9,** 19139–19160, doi: 10.1109/ACCESS.2021.3052311 (2021).

[26] Deepa, G. & Thilagam, P.S. "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," *Inf. Softw. Technol.*, **74,** 160–180, doi: 10.1016/J.INFSOF.2016.02.005 (2016).

[27] Banowosari, L.Y. & Gifari, B.A. "System Analysis and Design Using Secure Software Development Life Cycle Based On ISO 31000 and STRIDE. Case Study Mutiara Ban Workshop," *Proc. 2019 4th Int. Conf. Informatics Comput. ICIC 2019*, doi: 10.1109/ICIC47613.2019.8985938 (2019).

[28] Muzafar, S., Jhanjhi, N.Z., Khan, N.A. & Ashfaq, F. "DDoS Attack Detection Approaches in on Software Defined Network," *14th Int. Conf. Math. Actuar. Sci. Comput. Sci. Stat. MACS 2022*, doi: 10.1109/MACS56771.2022.10022653 (2022).

[29] Muzafar, S., & Jhanjhi, N. "DDoS Attacks on Software Defined Network: Challenges and Issues," *2022 Int. Conf. Bus. Anal. Technol. Secur. ICBATS 2022*, 2022-Janua, doi: 10.1109/ICBATS54253.2022.9780662 (2022).

[30] Khan, S.R., Rehman, I. & Malik, S.U.R. "The impact of test case reduction and prioritization on software testing effectiveness," *2009 Int. Conf. Emerg. Technol. ICET 2009*, 416–421, doi: 10.1109/ICET.2009.5353136 (2009).

[31] Farhan, A.R.S. & Mostafa, G.M. "A Methodology for Enhancing Software Security during Development Processes," *21st Saudi Comput. Soc. Natl. Comput. Conf. NCC 2018*, doi: 10.1109/NCG.2018.8593135 (2018).

[32] Winter, R.J. "Agile Software Development: Principles, Patterns, and Practices," *Perform. Improv.*, **53**(4), 43–46, doi: 10.1002/PFI.21408 (2014).

[33] Wäyrynen, J., Bodén, M. & Boström, G. "Security engineering and eXtreme Programming: An impossible marriage?," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, **3134,** 117–128, doi: 10.1007/978-3-540-27777-4_12/COVER (2004).

[34] Bostrm, G., Vayrynen, J., Boden, M., Beznosov, K. & Kruchten, P. "Extending xp practices to support security requirements engineering," *Proc. - Int. Conf. Softw. Eng.*, vol. 2006-May, 11–17, doi: 10.1145/1137627.1137631 (2006).

[35] Ge, X., Paige, R.F., Polack, F.A.C., Chivers, H. & Brooke, P.J. "Agile development of secure web

applications," *ICWE'06 Sixth Int. Conf. Web Eng.*, 305–312, doi: 10.1145/1145581.1145641 (2006).

[36] Mouratidis, H. & Giorgini, P. "Integrating security and software engineering: Advances and future visions," *Integr. Secur. Softw. Eng. Adv. Futur. Visions*, 1–288, doi: 10.4018/978-1-59904-147-6 (2006).

[37] Beznosov, K. & Kruchten, P. "Towards agile security assurance," *Proc. New Secur. Paradig. Work.*, 47–54, doi: 10.1145/1065907.1066034 (2005).

[38] Hoeren, T. & Pinelli, S. "Agile programming – Introduction and current legal challenges," *Comput. Law Secur. Rev.*, **34**(5), 1131–1138, doi: 10.1016/J.CLSR.2018.04.004 (2018).

[39] "Agile Security Issues: A Research Study General." https://www.researchgate.net/publication/267832772_Agile_Security_Issues_A_Research_Study_General (accessed Oct. 05, 2023).

[40] Khan, R. "Secure software development: a prescriptive framework," *Comput. Fraud Secur.*, **2011**(8), 12–20, doi: 10.1016/S1361-3723(11)70083-5 (2011).

[41] Thorpe, R. & Holt, R. "The SAGE Dictionary of Qualitative Management Research," *SAGE Dict. Qual. Manag. Res.*, doi: 10.4135/9780857020109 (2008).

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

[43] Afzal, W., Torkar, R. & Feldt, R. "A systematic review of search-based testing for non-functional system properties," *Inf. Softw. Technol.*, **51**(6), 957–976, doi: 10.1016/J.INFSOF.2008.12.005 (2009).

[44] Elijah, A.V., Abdullah, A., JhanJhi, N.Z., Supramaniam, M. & Balogun A.O., "Ensemble and Deep-Learning Methods for Two-Class and Multi-Attack Anomaly Intrusion Detection: An Empirical Study" International Journal of Advanced Computer Science and Applications(IJACSA), **10**(9). http://dx.doi.org/10.14569/IJACSA.2019.0100969 (2019).

[45] Sennan, S. et al. Energy efficient optimal parent selection based routing protocol for Internet of Things using firefly optimization algorithm. Transactions on Emerging Telecommunications Technologies, **32**(8), e4171 (2021).

[46] Gaur, L. et al. Disposition of youth in predicting sustainable development goals using the neuro-fuzzy and random forest algorithms. Human-Centric Computing and Information Sciences, **11,** NA (2021).

[47] Nanglia, S., Ahmad, M., Khan, F.A.W. & Jhanjhi, N.Z. "An enhanced Predictive heterogeneous ensemble model for breast cancer prediction." Biomedical Signal Processing and Control **72,** 103279 (2022).

[48] Gaur, L. et al. Capitalizing on big data and revolutionary 5G technology: Extracting and visualizing ratings and reviews of global chain hotels. Computers and Electrical Engineering, **95,** 107374 (2021).

[49] Kumar, T., Pandey, B., Mussavi, S.H.A. & Zaman, N. "CTHS based energy efficient thermal aware image ALU design on FPGA." Wireless Personal Communications **85,** 671-696 (2015).

[50] Gouda, W., Sama, N. U., Al-Waakid, G., Humayun, M., & Jhanjhi, N. Z. (2022, June). Detection of skin cancer based on skin lesion images using deep learning. In Healthcare **10**(7), 1183, MDPI (2022).

[51] Lim, M., Abdullah, A., & Jhanjhi, N. Z. Performance optimization of criminal network hidden link

prediction model with deep reinforcement learning. Journal of King Saud University-Computer and Information Sciences, **33**(10), 1202-1210 (2021).

[52] Hussain, K., Hussain, S. J., Jhanjhi, N. Z., & Humayun, M. SYN flood attack detection based on bayes estimator (SFADBE) for MANET. In 2019 International Conference on Computer and Information Sciences (ICCIS) 1-4. IEEE (2019).

[53] Hussain, I. et al. Health monitoring system using internet of things (iot) sensing for elderly people. In *2022 14th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)* 1-5. IEEE (2022).

[54] Zahra, F. et al. Rank and wormhole attack detection model for RPL-based internet of things using machine learning. *Sensors*, **22**(18), 6765 (2022).

[55] Humayun, M. et al. Software-as-a-service security challenges and best practices: A multivocal literature review. *Applied Sciences*, **12**(8), 3953 (2022).

[56] Humayun, M., Jhanjhi, N. Z., & Almotilag, A. Real-time security health and privacy monitoring for Saudi highways using cutting-edge technologies. *Applied Sciences*, **12**(4), 2177 (2022).

[57] Alotaibi, A. F. A comprehensive survey on security threats and countermeasures of cloud computing environment. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, **12**(9), 1978-1990 (2021).

[58] Kumar, K., Verma, S., Jhanjhi, N. Z., & Talib, M. N. A Survey of The Design and Security Mechanisms of The Wireless Networks and Mobile Ad-Hoc Networks. In *IOP Conference Series: Materials Science and Engineering* **993**(1), 012063. IOP Publishing (2020).

[59] Sangkaran, T., Abdullah, A., & JhanJhi, N. Z. Criminal network community detection using graphical analytic methods: A survey. *EAI Endorsed Transactions on Energy Web*, **7**(26), e5-e5 (2020).

[60] Almusaylim, A. Z., Jhanjhi, N.Z, & Alhumam, A. Detection and mitigation of RPL rank and version number attacks in the internet of things: SRPL-RP. *Sensors*, **20**(21), 5997 (2020).