AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 2, July, 2017, PP.304-316

REVIEW ARTICLE

Analysis of software security model in scenario of Software Development Life Cycle (SDLC)

Nor Shahriza Abdul Karim, Tanzila Saba, Arwa Albuolayan

College of Computer and Information Sciences, Prince Sultan University Riyadh 11586, Saudi Arabia.

Abstract: Software security has become an essential requirement for software systems. However, still software development methodologies do not pay enough attention to introduce security measures into their life cycles. Nonetheless, if security is considered in the SDLC, several security vulnerabilities that are evident in production would certainly disappear and the resolution costs would be much reduced since they will be detected and removed at the earlier stage in SDLC. The proposed software security model is evaluated with respect to security practices and their concerns among software engineers in various software development projects. Results are compared with reported software security techniques elaborated in categories along with their applications and limitations. Finally, suggestions are recommended to ensure software security in SDLC

Keywords: Secure Software Development; Secure Engineering; Software Security; SDLC

1. Introduction

Software security is one of the most important features in software development process that requires high attention among engineers. Indeed, all software faces threats from various potential malicious adversaries that are growing every day; from web mindful applications running on PCs, to complex media communications [1]. These threats and more can impose a huge challenge to software engineers in designing measures as part of their risk management activities as well as designing the appropriate security requirements and policies. This is due the degree of subjectivity in how security is being perceived and subject to different levels of concerns. Moreover, numerous software are developed without paying due attention to security issues such as confidentiality, integrity, access control, and non-repudiation [2].
Research gap could be found in many areas in software security [3,4] Pohl and Hof, 2015; Neamah et al., 2014). Agile methods are criticized as no security measures are taken during their development. One important area as noted by Devanbu and Stublebine (2000) is on nonfunctional requirements of a software security, which is not well focused in software security research. It takes on the topic involving effective design and use of policies and framework for security requirements in order to sufficiently address software security concerns in various software development projects (Mundher et al., 2014).

This research, shall therefore, explore security practices and concerns among engineers in various software development projects by focusing on each phase of the software development lifecycle. The result is expected to enhance software security practices with fewer defects and vulnerabilities, through common understanding on standards, policies, procedures and framework (Karim et al., 2016; Rehman and Saba, 2014).

2. Research Background

Effective software is one that is written to achieve its intended purpose and to ensure that it provides pleasant user experience in a completely safe environment. Often time, it is highly challenging to code software application that meets all the success criteria defined. (Sodiya et al., 2006). The big question mark is how could to write software that will be resilient to security flaws?

Engineers, often time, are less aware of security approaches leading to functional software but highly vulnerable to security threat at the end. This is likely to occur, due to various flaws in the approach to development when the focus is given on the functional role of the software, while ignoring many security concerns during the process. Many of them attempted to security aspects by revisiting the work later at the end (Gilliam., 2004).

Unsurprisingly, many software products undergoing live testing are vulnerable to threats and fail to provide a safe environment to the users and clients. This is probably due to lack of systematic measures such as reviews, checks and procedures or a framework, that could assist developers and managers of projects ensure that security processes in software development are followed according to a set of rules or procedures consistently throughout the development process (Nunes et al.,2010). Therefore, an approach or a framework is needed to build security measures, right from the beginning and consistently followed throughout the process (Gilliam et al., 2003). Mouratidis et al., (2005) has reported that "software engineers consider security as a non-functional requirement, but unlike other non-functional requirements, like reliability and performance, security has not been fully integrated within the development lifecycle". However, it is only considered after the design of the system is completed or when an issue of vast concern rose.

In avoiding issues of security that exist in many development projects, it is considered necessary to incorporate security-minded thinking throughout the development process. As such, missing important security requirements, or making critical mistakes in the software design could be avoided. This way, problems could be discovered much earlier instead of at the end when it is very hard or costly to fix (Jones and Rastogi, 2004; Ahmad et al., 2014). Vulnerabilities associated with the system may be reduced to minimum level during SDLC. It may not be possible to eliminate vulnerability but it might be reduced to the minimum level because it is an ongoing process (Ahmad et al., 2015).

Software security is one of the most important features in software development that requires high attention among engineers. Indeed, all software faces threats from various potential malicious adversaries that are growing exponentially (Devanbu and Stubblebine, 2000). These threats could impose a huge challenge to software engineers in designing measures as part of their risk management activities as well as designing the appropriate security requirements and policies. This is due the degree of subjectivity in how security is being perceived and subject to different levels of concerns. Additionally, numerous applications have been created without considering security issues, for example, classification, uprightness, get to control, and non-renouncement (Shin and Gomaa, 2007).

Research gap could be found in many areas in software security (Pohl and Hof, 2015). "The exclusion of security elements from the agile development process creates vulnerable software. This leads to reiteration in order to make the software secure. This affects the project timeline, raises costs, and negatively affects customer satisfaction, which ultimately diminishes the notion of such a methodology being agile" (Arbain et al., 2014). One important area as noted by Devanbu and Stublebine (2000) is on non-functional requirements of a software security, which is not well focused in software security research. It takes on the topic involving effective design and use of policies and framework for security requirements in order to sufficiently address software security concerns in various software development projects.

This research, therefore, explore security practices and concerns among engineers in various software development projects by focusing on each phase of the software development lifecycle. The result is expected to enhance software security practices with fewer defects and vulnerabilities, through common understanding on standards, policies, procedures and framework.

Additionally, the current research provides an opportunity to enhance our understanding in the area of secure software development process in the contact of the case and SDLC. Moreover, research is also expected to assist in the development of a practical guide for secure software development process. This effort will also bring together an analysis to some of the international standards and best practices with a study of the practices and choices made by teams of software engineers (Alkawaz et al., 2016).

3. Importance of Secure Software Engineering

According to GARTNER reports (Brodkin, 2008), most of the attacks are targeting the application layer. Therefore, software security defects are a major concern for security professionals to deal with. Security problems are complicated. They can be implementation bugs or architectural flaws (McGraw and Hoglund, 2004). This tendency pushed researchers to come out with embedded security in software development lifecycle.

Mcgraw (2004) had briefly discussed engineering software with the goal that it keeps on working effectively under malignant assault. Still, engineers needs some help in understanding how to tackle it as the software security field is a relatively new one. Additionally, the author explained why software security must be part of a full lifecycle approach since that a security issue will probably emerge as a result of an issue in a standard-issue part of the framework than in some given security feature. Pretty much as you can't test quality in a bit of programming, you can't splash paint security highlights onto a plan and anticipate that it will get to be secure. Gilliam et al., (2003) highlighted the importance of integrating security as a formal approach in the software lifecycle. It is essential to protect corporate resources; nevertheless, little thought has been given to this aspect since software security has been treated as an afterthought. As a result, they proposed a Software Security Checklist to be followed by software engineers (Gilliam et al., 2003; Fadhil et al., 2016).

As per Sommerville (2010), it is exceptionally hard to add security to a framework after it has been actualized. Along these lines, authors added the need to consider security issues amid the frameworks outline handle and clarified application-autonomous issues pertinent to secure frameworks plan. One is the way do compositional outline choices influence the security of a framework. Second, what is acknowledged great practice when outlining secure frameworks. Third, what support ought to be planned into frameworks to maintain a strategic distance from the presentation of vulnerabilities when a framework is conveyed for utilize. Fischer and Spada (2014) described the approach for standardization and addressing secure software engineering in the European Space Agency (ESA). Secure and vigorous operational software is turning out to be increasingly basic perspective in data security to alleviate section focuses for programmers and digital offenders. ESA is resolved to enhance the security of its operational programming by applying secure programming designing philosophies (Rehman et al., 2014).

4. Secure SDLC and Current Research

A few researches could be found in the related area. Works are either at the conceptual level or related for to functional requirements. Some works like Mcgraw (2004), Wongthongtam et. al. (2009), characterized what should be included as part of the security requirements, and others like Islam and Falcarin (2011), Pohl and Hof (2015), stressed on technical issues and methodology.

Developing software with good quality is a hard and complex task. It involves additional security-dedicated activities that are usually omitted in traditional SDLC (Essafi and Ghezala, 2014). As most developers are not trained for software security, Daud (2010) also stressed on the importance of security measures at each phase of SDLC. The proposed model suggests that security engineers should elicit security requirements using different methods. Once these requirements are refined, then design phase can starts. Security vulnerabilities are identified and documented to be passed to development team. Developers will take into consideration all vulnerabilities and their mitigation plan designed during design phase. Testing team then should receive the developed software along with the documentation in order to find design and development bugs. At the end, software is ready for deployment.

Bokhari and Siddiqui (2014) proposed a tool for secure software requirement management. The tool is known as TSSR (Tool for Secure Software Requirements). Security testing should cover two strategies: one is testing security functionality with functional testing techniques, and the second is a risk-based security testing that is based on attack patterns and threat models (Wongthongtham et al., 2009). A security test plan is set with traceability back to requirements and employs both strategies that could be considered as a good testing plan. Penetration testing is useful as well, if it considers the software architecture. In another research, Mcgraw (2009) built up an assemblage of ten best practices for secure programming improvement that mirror the experience and ability of a few partners of the product advancement lifecycle (SDLC).These stakeholders include software engineers, auditors, operational personnel and management. Islam and Falcarin (2011) found that safe programming is about moderating risks to accomplish the business objectives and that security relies on upon the setting where programming is sent. In fact, there is no valid yardstick to measure security legitimately characterized for measuring programming security. They proposed how to identify security requirements through asset based risk management process in order to describe software security goal. Then based on the proposed approach they can evaluate the security requirements to measure software security (Islam et al., 2011).

Russell et al., (2004) stressed on the importance of testing and its role to verify that the system design and code can resist attack. A security test plan is essential to produce secure software. It should validate that software meets the security requirements. Additionally, serious attempts to penetrate and break software security and a scan for common vulnerabilities should be included.

McGraw (2003) found that software practitioners are becoming aware of software security issues, and more work is to be done in this area. Pohl and Hof (2015) highlighted on the increase use of agile software development methods and the current attack landscape. Hence, developing secure software should be a main concern in all software development projects. In short, not much research could be found in the related area. Works are either at the conceptual level or related for to functional requirements. Some works like Mcgraw (2004), Wongthongtam et. al. (2009), characterized what should be included as part of the security requirements, and others like Islam and Falcarin (2011). Bokhari and Siddiqui (2014) proposed a tool for secure software requirement management. Pohl and Hof (2015), stressed on technical issues and methodology, in addition to the concept of secures software development practices, tools are designed to ensure effectiveness of security measures at some stage in the process.

5. International Standards for Secure Software Development

Nunes et al., (2010) reported that paying little mind to all speculations made in process change, there is still no certification that the created programming is shielded from assaults or it doesn't present security vulnerabilities. Still, measures, for example, Capability Maturity Model (SSE-CMM) and global guidelines, for example, ISO/IEC 15408 ought to be considered to bolster the advancement of secure programming (Herbsleb et al., 1997). Mesquida and Mas (2015) built up a careful mapping between the ISO/IEC 27002 security controls and the ISO/IEC 15504-5 base practices for programming lifecycle forms. As they found that it is a need for organizations to guarantee its coherence, minimize vulnerabilities and expand the arrival of speculation and business openings. Correspondingly, a noteworthy number of programming organizations, that have been included in a procedure change program as per ISO/IEC 15504 (ISO/IEC,2004b, 2004c), request the usage of ISO/IEC 27000 as a security standard (Mesquida and Mas, 2011). A list of international standards that addresses the security practices in software development are exhibited in Table 1.

Table 1 Overview of Existing Software Security Standards

Standard	**Description**
ISO/IEC 15408	The Common Criteria – ISO/ IES 15408 contains criteria for evaluation of IT Security that are broadly useful within the international community (Nishimura et al., 2005).
ISO/IEC 27001	A globally perceived best practice structure for a data security administration framework. It helps associations distinguish the dangers to critical data and set up the suitable controls to lessen the hazard (Humphreys, 2006).
SSE-CMM	Describes the essential characteristics of an organization's security engineering process that must exist to ensure good security engineering (Yang et al., 2015).
ISO/IEC 21827	The ISO/IEC 21827 is an International Standard that is based on the Systems Security Engineering Capability Maturity Model (SSE-CMM) and it help organizations to identify security goals, assess security posture, and support security life cycle (Mellado et al., 2007).
OWASP Top Ten	A rundown of the 10 Most Critical Web Application Security Risks. What's more, for every risk it gives: a portrayal, illustration vulnerabilities, case assaults, direction on the most proficient method to maintain a strategic distance from (OWASP, 2010).

6. Benchmark Frameworks

6.1 Iterative Model

Daud (2010) proposed iterative model to concentrate on security prerequisites at every stage of programming life cycle. The study concentrated on Extreme programming (XP) approach. What's more, the referred to research is a guide for designers to create secure programming as the greater part of the product engineers are not prepared for programming security. Mix of security and XP give another approach that is appeared in figure 1.

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 2, July, 2017, PP.304-316

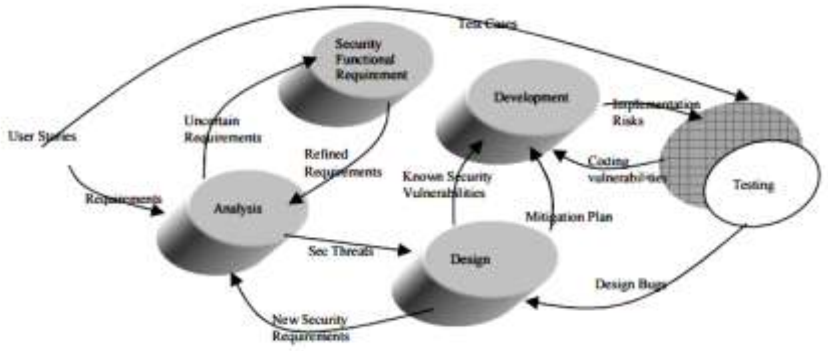

Figure 1: Iterative Model of Secure Software Life Cycle

Once the indeterminate prerequisites are refined by security useful necessity (SFR) module, outlining the product begins. Plan stage is imperative and requires more thought as far as security. In light of the data gave by investigation stage (Security Requirements by client stories and SFR) a danger model is created. On the off chance that security build feels a portion of the data is absent or some other security dangers are conceivable then it does a reversal to investigation for the refinement of the security prerequisites. In the event that security master finds no issues, then a relief plan is intended to provide food each one of those dangers recorded in risk show. Security vulnerabilities are distinguished amid outline stage. Every one of the vulnerabilities that a product framework may experience the ill effects of are recorded and go to advancement group. Engineers begin advancement by considering all vulnerabilities and their relief arrange planned amid outline stage. When programming is created then it is given over to testing group alongside the documentation. Necessities designing are the primary building obstruct for any product improvement. Security engineers attempt to evoke security prerequisites by various strategies, e.g. client stories, manhandle cases, and so forth. Figure 2 exhibits principle operation to be performed amid SSLC.

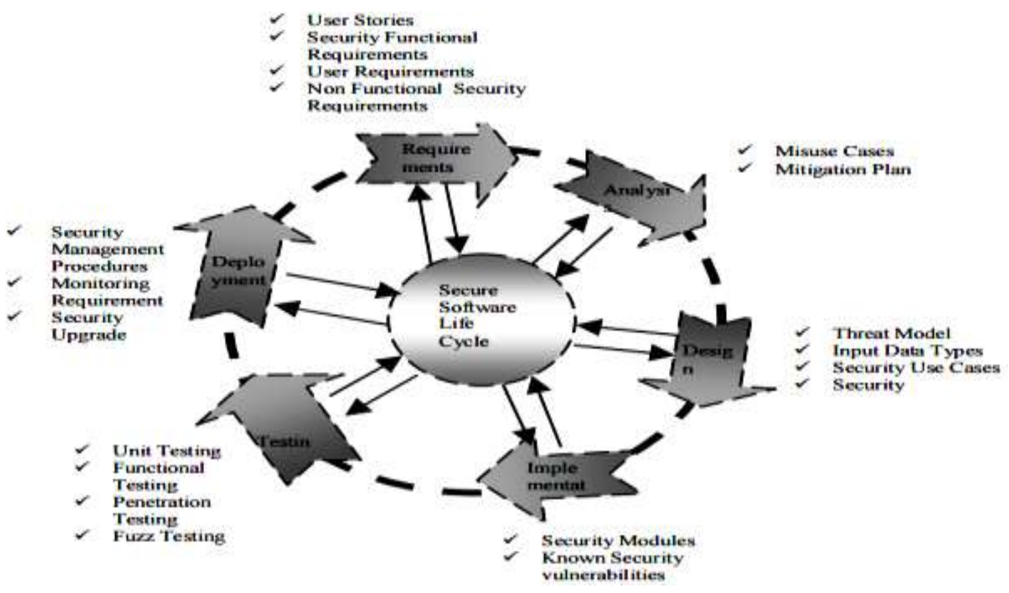

Figure 2 Secure Software Life Cycle

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 2, July, 2017, PP.304-316

Figure 2 demonstrates all changes in the programming life cycle to ensure security in programming. Actually, this life cycle is iterative in nature. In the event that few security modules are not considered at any stage of life cycle, then architects could retraced to that stage and satisfy those deficiencies.

6.2 Prescriptive Framework

A prescriptive framework was proposed by Khan and Zulkernine (2009). Figure 3 shows the steps involved during requirement phase.

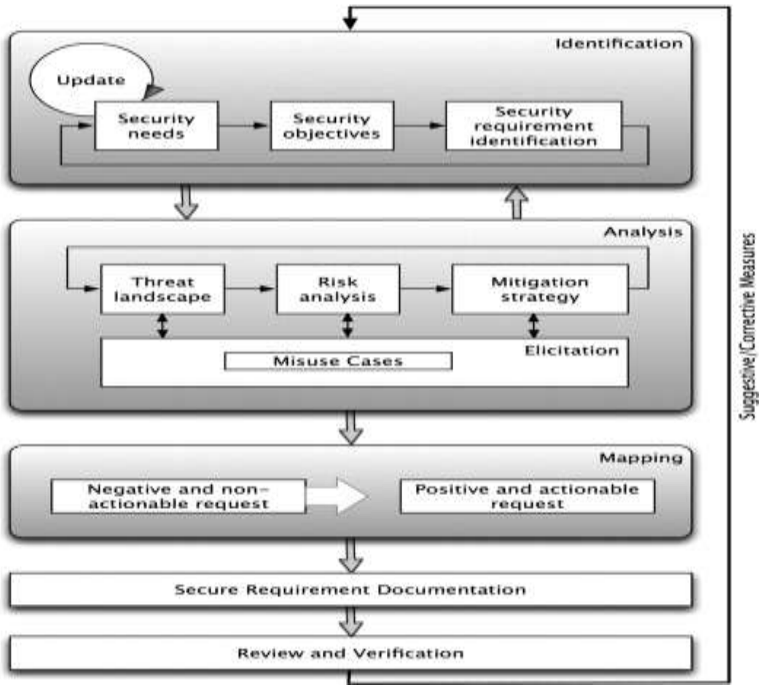
Identification
Update
Security needs
Security objectives
Security requirement identification
Analysis
Threat landscape
Risk analysis
Mitigation strategy
Elicitation
Misuse Cases
Mapping
Negative and non-actionable request
Positive and actionable request
Secure Requirement Documentation
Review and Verification
Suggestive/Corrective Measures

Figure 3. Secure requirement specification

- Review and Verification: Review documentation against the objectives and needs.

i. Design phase

Figure 4 shows the steps involved during this phase.

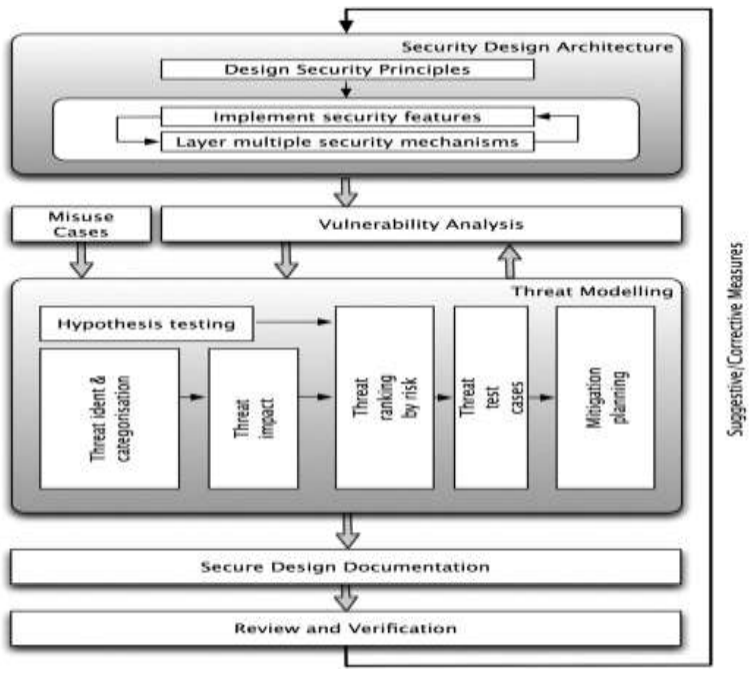
Security Design Architecture
Design Security Principles
Implement security features
Layer multiple security mechanisms
Misuse Cases
Vulnerability Analysis
Threat Modelling
Hypothesis testing
Threat ident & categorisation
Threat impact
Threat ranking by risk
Threat test cases
Mitigation planning
Secure Design Documentation
Review and Verification
Suggestive/Corrective Measures

Figure 4. Secure Design

ii. Coding Phase: Prescriptive exercises in every progression amid secure coding:

• Secure code composing: documentation and rules created amid before stages could control software engineers in composing secure code.

• Static investigation: source code examination devices are currently turning out to be exceptionally basic, and the utilization of such apparatuses is profoundly suggested.

• Code Review: secure code audits could be manual or automatic.

iii. Testing Phase: Amid security testing, the analyzer by and large uses abuse cases, danger models and outline records to recognize conceivable assaults and the effects of fruitful assaults.

• Prepare security test cases: in this progression, test cases are intended to assault the product.

• Security testing documentation: in the wake of performing security testing, test records are readied comprising of security experiments and an organized rundown of vulnerabilities.

iv. Development stage: After the product is conveyed into its operational surroundings, it is critical to screen reactions to defects and vulnerabilities of the framework to check for new advanced examples. In the wake of recognizing new security designs, the same ought to be incorporated into the necessity organize for further security enhancements in consequent discharges. The safe sending stage includes last security surveys and reviews.

v. Maintenance stage: Before organization, heads must comprehend the security issues of the product. After arrangement, a portion of the distinguished blemishes that were not tended to already will be taken a gander at once more, organized and altered. In this stage, new threats are observed.

7. Security stage

Haridas (2007) proposed an evolving solution (Figure 5) that should be considered throughout SDLC phases. It should be carried along with other activities ideally followed in SDLC.

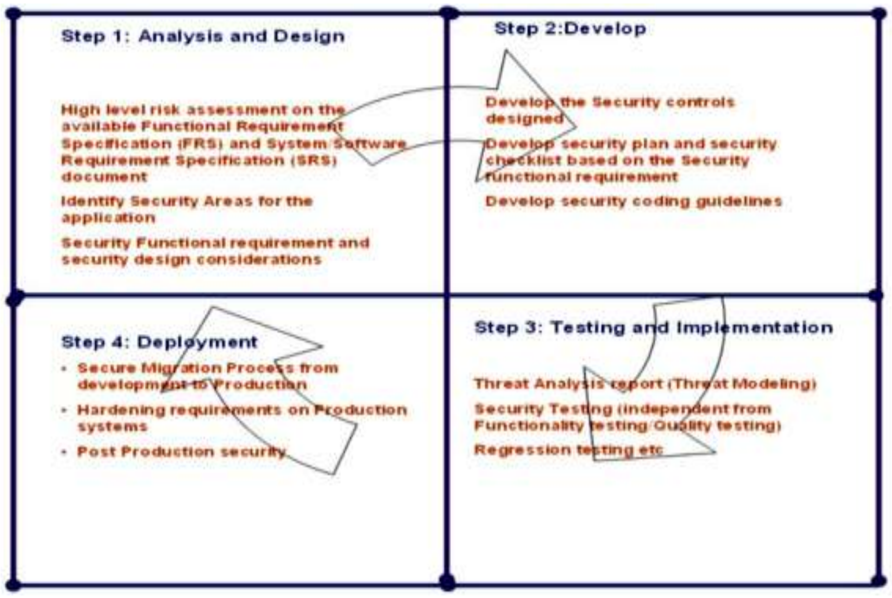
Step 1: Analysis and Design
High level risk assessment on the available Functional Requirement Specification (FRS) and System Software Requirement Specification (SRS) document
Identify Security Areas for the application
Security Functional requirement and security design considerations
Step 2:Develop
Develop the Security controls designed
Develop security plan and security checklist based on the Security functional requirement
Develop security coding guidelines
Step 3: Testing and Implementation
Threat Analysis report (Threat Modeling)
Security Testing (independent from Functionality testing Quality testing)
Regression testing etc
Step 4: Deployment
Secure Migration Process from development to Production
Hardening requirements on Production systems
Post Production security

Figure 5: Security in SDLC Phases

8. Conclusion and suggestions

This research has presented an overview of the issues involved in software security and in the software development process. As of now, no software improvement techniques or practices exist that dependably make secure software development. It is consequently, suggested that designers receive practices that could reduce software defects and as a result, minimize any potential risk due to lack of security attention during the process. Hence, it is further suggested that all security measures should be implemented from start in SDLC to ensure secure environment and user's satisfaction.

References

Ahmad, Z., Asif, M., Shahid, M., and Rauf, A. (2015). Implementation of Secure Software Design and their impact on Application. *International Journal of Computer Applications*, 120(10).

Ahmad, AM., Sulong, G., Rehman, A., Alkawaz,MH., Saba, T. (2014) Data Hiding Based on Improved Exploiting Modification Direction Method and Huffman Coding, Journal of Intelligent Systems, vol. 23 (4), pp. 451-459, doi. 10.1515/jisys-2014-0007.

Alkawaz, M.H., Sulong, G., Saba, T. Rehman, A (2016). Detection of copy-move image forgery based on discrete cosine transform, Neural Computing and Application. doi:10.1007/s00521-016-2663-3.

Arbain, A. F., Ghani, I., and Jeong, S. R. (2014). A Systematic Literature Review on Secure Software Development using Feature Driven Development (FDD) Agile Model, 15(1), 13-27.

Bokhari, M. U., and Siddiqui, S. T. (2014). TSSR: A Proposed Tool for Secure Software Requirement Management. *International Journal of Information Technology and Computer Science (IJITCS)*, *7*(1), 1.

Brodkin, J. (2008). Gartner: Seven cloud-computing security risks. InfoWorld, July, 2–3. Retrieved from http://www.idi.ntnu.no/emner/tdt60/papers/Cloud_Computing_Security_Risk.pdf

Daud, M. I. (2010, March). Secure software development model: A guide for secure software life cycle. In *Proceedings of the international MultiConference of Engineers and Computer Scientists* (Vol. 1, pp. 17-19).

Devanbu, P. T., and Stubblebine, S. (2000, May). Software engineering for security: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 227-239). ACM.

Essafi, M., and Ghezala, H. B. (2014). Meta-Modeling Based Secure Software Development Processes. International Journal of Secure Software Engineering (IJSSE), 5(3), 56-74.

Fischer, D., and Spada, M. "Ready for Secure Software: Secure Software Engineering for Space Missions" , *Spaceops 2014 conference*.

Fadhil, MS. Alkawaz, MH., Rehman, A., Saba, T. (2016) Writers identification based on multiple windows features mining, 3D Research, vol. 7 (1), pp. 1-6, doi.10.1007/s13319-016-0087-6

GARTNER. Available: www.gartner.com.

Gilliam, D. P. (2004). Security risks: management and mitigation in the software life cycle. 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.

Mundher, M. Muhamad, D. Rehman, A. Saba, T. Kausar, F. (2014) Digital watermarking for images security using discrete slantlet transform, Applied Mathematics and Information Sciences, vol 8(6), pp. 2823-2830, doi.10.12785/amis/080618.

Gilliam, D. P., Wolfe, T. L., Sherif, J. S., and Bishop, M. (2003, June). Software security checklist for the software life cycle. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on* (pp. 243-248). IEEE.

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., and Paulk, M. (1997). Software quality and the capability maturity model. *Communications of the ACM*, 40(6), 30-40.

Humphreys, T. (2006). State-of-the-art information security management systems with ISO/IEC 27001: 2005. *ISO Management Systems*, *6*, 1.

Neamah, K. Mohamad, D. Saba, T. Rehman, A. (2014). Discriminative features mining for offline handwritten signature verification, 3D Research vol. 5(2), doi. 10.1007/s13319-013-0002-3

Islam, S., and Falcarin, P. (2011, September). Measuring security requirements for software security. In *Cybernetic Intelligent Systems (CIS), 2011 IEEE 10th International Conference on* (pp. 70-75). IEEE.

Islam, S., Mouratidis, H., and Jürjens, J. (2011). A framework to support alignment of secure software engineering with legal regulations. Software and Systems Modeling, 10(3), 369–394

Jones, R. L., and Rastogi, A. (2004). Secure coding: building security into the software development life cycle. *Information Systems Security*, *13*(5), 29-39.

Karim, N.S.A, Albuolayan, A., Saba, T. and Rehman, A (2016). The practice of secure software development in SDLC: an investigation through existing model and a case study, *Security Comm. Networks*, doi. 10.1002/sec.1700.

Khan, M. U. A., and Zulkernine, M. (2009). Activity and artifact views of a secure software development process. Proceedings of 12th IEEE International Conference on Computational Science and Engineering, CSE 2009 (Vol. 3, pp. 399–404).

McGraw, G. (2003). Building Secure Software: A difficult but critical step in protecting your business. *Digital, White Paper, available at: http://www. digital. com/whitepapers*.

McGraw, G. (2004). Software security. *Security and Privacy, IEEE*, *2*(2), 80-83.

McGraw, G., and Hoglund, G. (2004, August). Exploiting Software: How to Break Code. In Invited Talk, Usenix Security Symposium, San Diego.

Mesquida, A. L., and Mas, A. (2011). ISO/IEC 15504-5 best practices for IT service management. In Communications in Computer and Information Science (Vol. 172, pp. 14–24).

Mesquida, A. L., and Mas, A. (2015). Implementing information security best practices on software lifecycle processes: The ISO/IEC 15504 Security Extension. *Computers and Security*, *48*, 19-34.

Mellado, D., Fernández-Medina, E., and Piattini, M. (2007). A common criteria based security requirements engineering process for the development of secure information systems. *Computer standards and interfaces*, *29*(2), 244-253.

Mouratidis, H., Giorgini, P., and Manson, G. (2005). When security meets software engineering: a case of modelling secure information systems.*Information Systems*, *30*(8), 609-629.