American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)
ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

http://asrjetsjournal.org/

# A Review of Security Issues in SDLC

Nosheen Nazir[a*], Muhammad Kashif Nazir[b]

[a,b]*Riphah College of Computing, Riphah International University, Faisalabad Campus, Punjab, Pakistan*
[a]*Email: nosheennazir304@gmail.com*
[b]*Email: kashifntuf@gmail.com*

**Abstract**

Software Engineers do not implement security as a continuing process in software development; they give it worth at the end of software development. Security implementation is an essential on-going routine in each phase of the software development lifecycle. This quantitative type of research investigates the security factors in different phases of Software Development Life Cycle (SDLC) and evaluates them from the research community and software engineers. Results are analyzed by using a statistical tool (SPSS), and security rules are proposed in each step of SDLC to assist software engineers and research community.

***Keywords:*** Security in SDLC; Review of SDLC; Security Rules in Software Development.

## 1. Introduction

The use of Information Technology has been increased day by day in business organizations. E-Business is a worthy way to compete with other organizations. The customers want to access the running applications in their industry at any time and from anywhere [1]. To meet this vision, applications are created in such a way; as to incorporate the necessary capability of ease in access as well as escorting multiple functionalities. Developers develop the required applications with faster rate and value added functionalities to support ever increasing and changing business needs to satisfy management. Therefore, the aspect of security implementation in all the phases of SDLC cannot be excluded [2].A software development is an intricate process and it meets the success criteria, if its intended purpose is achieved on a given platform by ensuring its requirement of security factors. There is a captivating need to define how a Software can be ductile to different security flaws in distinctive environments [1]. There can be many causes behind software security rifts, but most of the time, blame relies on intrusion of virus, denial of service and spam in email etc [3]. But if intense deep analysis is carried down, the fact undergo is a security breach that leads to defective, substandard and unreliable software which is vulnerable to virus and malicious attacks [4].

---

* Corresponding author.

Conventional SDLC applications pay emphasis to security concerns in the phase of testing; resultantly flaws in the system remain uncovered till its proper execution and use [5] It is an obligatory need to put our maximum efforts for implementation of security measures in software's each phase of Software Development Life Cycle (SDLC). A more secure, robust, reliable and free of virus-attacks software can be developed by focusing on security factors of SDLC that are discussed in this paper.

Why developers omit security measures in software development? The reason is, they give less importance to security factors and focus on timely development and delivery of software. Because, for considering security features in SDLC it takes time and leads to project delays [6, 7].

So, the designers themselves and instructed by Project managers, ignore the security features implementation [7]. However, the implementation of security features in SDLC will increase the software life as much from earlier and fulfills software's non-functional requirement of reliability by providing protection of data against losses due to security black holes and virus attacks etc. [8]Anyway, if virus attack is encountered in software, the system should possess the capability to recover its loss components and formulate some mechanism to avoid such kind of malicious attacks [3, 9].

Another group of researchers believe that security features have secondary importance; they consider it as a non-functional requirement, and like other non-functional requirements i-e. reliability and performance etc. it can be taken into consideration after the completion of development of software application[1, 10].

But security features have some contrariness from others they must be subscribed at initial of the project requirements gathering and continues with all phases of SDLC till testing and maintenance [11].

There is need to encourage people with such thinking that should lay stress on better understanding towards security policies, in the team from project manager to lower level developers to add security features at each level of SDLC to increase security.

Security is the basic need of today's software's and it is not the factor that should be considered once. Implementation of security should be taken into consideration from the very initial stages of software development [4] It is a continuous process that is applicable to all stages of SDLC i-e.

Requirement, Design, Development, and Testing to deal with vulnerabilities and malicious attacks [12]. By adopting security policies, the system can be made robust to deal with uncertain situations in an adverse environment and another non-functional requirement i-e.

Recoverability has roots in robust security [13]. A system is considered to be secure if its implementation is wholly correlated to its defined security policies [10, 14] Software is regarded as secure if data loss in any uncertain situation is recoverable and could be reported to deal with unusual situations[13, 15].

In this paper, the importance of security features has been discussed and deliberately taken into consideration and provided a review of existing literature to take security measures and polices from its initial stages and

continue them till the completion of development.

Also, the state of the art measures are presented with respect to SDLC phases and a complete guideline is suggested for the development of resilient software's that performs efficiently in even adverse environments.

## 2. SDLC Security And Related Research

### *2.1. SDLC Enlargement*

In the past, Developers focused on functionality related aspects of applications and intended to develop applications without following accurate and standardized engineering principles.

This was not a persuasive approach to develop large projects that need to be incorporated into significant engineering principles[16]. The main objectives of these principles are discussed below:

- Well Formulated Requirement's
- Well Oriented Software Development life cycle
- Well defined software development models

The organizations from several years manage SDLC documentation properly and the typical SDLC is as follows

**Table 1:** SDLC

| Function | Description |
|---|---|
| Conceptual Definition | Understanding of proposed project and objectives |
| Functional Requirements and specifications | List of business requirements |
| Technical Requirements and specifications | Detailed description of technical requirements |
| Design | Formal detailed design of the program developed |
| Coding | Development phase |
| Test | The testing phase of working product |
| Implementation | Deployment of project |

The last phase maintenance is omitted from the table because during the maintenance the whole Software Development Life Cycle is repeated. The validation present in the first cycle is needed every time thereafter.

### *2.2. Security Incorporation in SDLC*

Many researchers debate the importance of security factors in SDLC and result is the security features must be incorporated in each phase of SDLC.

Reference [17] Proposed a tool for software security which includes four requirements. 1. Planner: that used in analyzing software security and to investigate and analyze errors. 2.

Modeler: Framework demonstration 3. Prover: proficient thinking 4. Documenter: structural views and technical document editing. Reference [3] incorporate security in the following steps

**Table 2:** Different Security Rules

| Function | Description |
|---|---|
| Awareness | Update existing knowledge by accruing new information |
| Prevention | It describes the security synchronization in software |
| Accountability | Maintains a log file for all the tasks |
| Availability | An unauthorized person cannot access information |
| Integrity | An unauthorized person cannot alter the information |
| Availability | The highly secure system should be available all the time |
| Non-Repudiation | The transaction cannot be denied by any third party |
| Access Control | Resources cannot be accessible without permission |
| Accuracy | All tasks should be performed with 100 % accuracy. |
| Identification, Authentication | An only authenticated person can be logged into the system. |
| Authorization | What a subject can do on the system |
| Consistency | All security features are consistent |
| Assessment, Evaluation | Continuous assessment and evaluation of all software processes is necessary |
| Privacy | Individuals should know what information is collected about them and for what purpose |
| Flexibility | Flexibility reduces the rigid boundaries of software processes |
| Excellence | Security is a factor of quality |
| Unambiguity | Detail about everything should be clear and persistent |
| Fortification | Processes used in software should be secured individually and in totality |
| Auditability | Auditability is to judge the security of software |
| Error Classification | Errors should be classified as per schema containing a set of security rules |
| Interpretability | All software's must be secured by participating in any communication or transaction |

Security issues with respect to SDLC phases are discussed in Table 3:

**Table 3:** Security In SDLC

| Function | Description |
|---|---|
| Requirement | Elicit requirements from all stakeholders that are involved in the project directly or indirectly[18, 19]<br>Adopt international standards that fit into your organization [1] |
| Design | Maintain a list of recommended software's [2]<br>External review of the design [20] |
| Implementation | Follow secure coding checklist [21]<br>Use language that is more immune [18]<br>Train developers to develop defensible applications [18]<br>Conduct secure code assessments [22] |
| Testing | Test plan and development plan for the next test[23]<br>Security testing plan [24]<br>Conduct penetration testing [25]<br>Using statistical analysis tools [26]<br>Define the acceptable level of vulnerabilities [1] |

## 3. Methodology

This is a quantitative type of research in which data is collected regarding different rules about the Software Development Life Cycle in software (SPSS) from different resources of literature review discussed. Performed

cleansing, made questionnaire, evaluation of questionnaire from the research community, combined results, and proposed rules to support research community and software engineers about security measures in SDLC.

### *3.1. Data Collection*

Primarily, a collection of the rules is done that is presented in the literature review concerning security incorporation in different phases of SDLC. These rules are presented in Table 1,2,3[2-4, 27, 28]

### *3.2. Data Cleansing*

Data cleansing is the process of removing irrelevant attributes from the data collected from the literature review. It eliminates all those extraneous values that cause no harm to the Software Development Life Cycle process. Researchers proposed research, only those security measures that affect the security concerns in SDLC are 3.3. presented in the questionnaire.

### *3.3. Prepare Questionnaire*

After data cleansing step, a questionnaire is developed and is shown presented in Figure. This questionnaire put forth light on those security rules that are necessarily concerned with security factors with respect to all the phases of SDLC. Veritable and exact rules are depicted accompanying the true meaning of proposed rules is explained as well as their incorporation into the SDLC phases is explained thoroughly.

### *3.4. Evaluation*

Evaluation (Online + Hardcopy)

To evaluate this questionnaire, the following options were added to each question. 1. Strongly Agree 2. Agree 3. Neutral 4. Disagree. The only one option as the answer could be selected by the respondents against each question with a restriction of answering all the questions.

The gathered responses were collected through online google forms and through spreading hardcopy of these forms among the targeted respondents.

This quantitative type data is accumulated from distinctive sources e.g. software houses, universities etc., as they are located in geographically distinct places.

### *3.5. Analysis by SPSS*

SPSS Statistical tool is utilized for analysis in this research[29]. The collected responses from different resources were inserted into the SPSS and graphs were generated and presented in the Results and Discussions section below. *Rules Generation for Security in SDLC*

After doing this all research specify some rules presented in SDLC that have highest priority and likeness in

different software houses.

These rules incorporate the highest security in SDLC that will make the software resilient to face security challenges from different intruders and attackers [29]

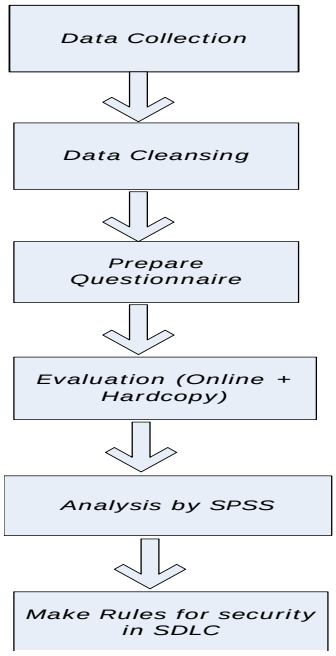


**Figure 1:** Methodology

## 4. Results and Discussions

After deep analysis of questionnaire (online+ hard copy), it was unearthed that only 5% respondents disagreed, 13% were just agreed, about 6% respondents were neutral, and approximately 70% of respondents strongly agreed in favor of the proposition that element of security shall be considered as the paramount factor in each phase of SDLC as shown in Figure 6.

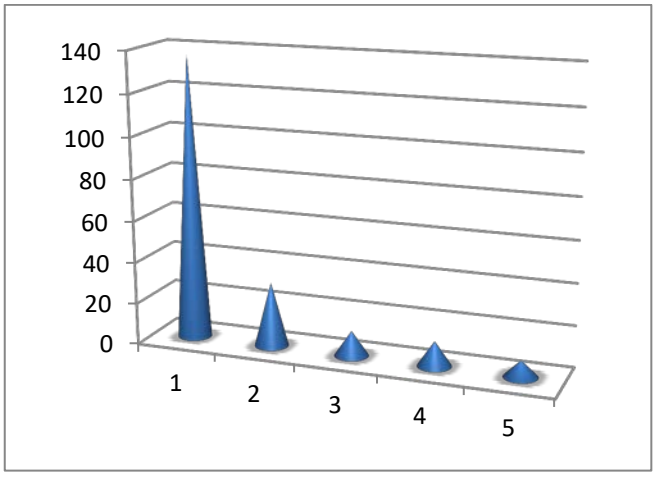


**Figure 2:** Requirement Phase Acceptance

**Table 4:** Security Incorporation in SDLC

| Phase | Rule | Rule Description |
|---|---|---|
| Requirement | Stakeholders identification | Proper stakeholder's identification incorporates security in the initial phase of requirement gathering. [3] |
| | Requirement Elicitation | Elicit requirement from all stakeholders with respect to their authorities and responsibilities.[18] |
| | Remove Conflicts | A conflict may arise in some requirements. Assess the respective stakeholder and priorities it on lower authority stakeholder. |
| | Stakeholders Satisfaction | Satisfy stakeholders whose requirements are satisfied on higher level decisions. |
| | International Standards | Adopt international standards that fit into your organization. [1] |
| | Threats identification | Adopt this process to identify and remove threats in requirement phase. |
| | Prevention | Security synchronization in each phase of SDLC described in this phase. [3] |
| | Unambiguity | Remove ambiguous requirements [3] |
| Design | Recommended Software's | Maintain a list of recommended software's framework [2] |
| | Security Principles | Explicitly apply security principles to the design. [30] |
| | External Review | External review of the design. [20, 24, 25] |
| | Access Control | Define access controls in design phase. |
| Development | Check List | Follow secure coding checklist. [31] |
| | Language | More immune language usage increase security. [18] |
| | Defensible | Train developers to produce defensible applications.[18] |
| | Assessment | Conduct source code assessment process. [26] |
| | Pair Programming | Use of pair programming reduces security risks.[32] |
| | Flexibility | Do not create rigid boundaries make it flexible to face security challenges in the present and future. |
| Testing | Testing plan | Security testing plan.[1] |
| | Analysis Tools | By using static analysis tools. [26] |
| | Acceptance Level | Define the acceptance level of vulnerabilities within coding and testing stages [1] |

**Table 5:** Requirement Phase Acceptance

| Requirement Phase Rule | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree | Total |
|---|---|---|---|---|---|---|
| Stakeholder Identification | 15 | 7 | 2 | 1 | 0 | 25 |
| Requirement Elicitation | 14 | 5 | 1 | 3 | 2 | 25 |
| Remove Conflicts | 18 | 5 | 2 | 0 | 0 | 25 |
| Satisfy Stakeholders | 20 | 2 | 2 | 1 | 0 | 25 |
| International Standards | 11 | 9 | 1 | 3 | 1 | 25 |
| Threats Identification | 16 | 1 | 2 | 2 | 4 | 25 |
| Prevention | 20 | 2 | 1 | 1 | 1 | 25 |
| Unambiguity | 23 | 0 | 1 | 1 | 0 | 25 |
| Total | 137 | 31 | 12 | 12 | 8 | 200 |
| %age | 68.5% | 15.5% | 6% | 6% | 4% | |

**Table 6:** Design Phase Acceptance

| Design Phase Rule | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree | Total |
|---|---|---|---|---|---|---|
| Recommended Software's | 15 | 7 | 2 | 1 | 0 | 25 |
| Security Principles | 14 | 5 | 1 | 3 | 2 | 25 |
| External Review | 18 | 5 | 2 | 0 | 0 | 25 |
| Access Control | 20 | 2 | 2 | 1 | 0 | 25 |
| Total | 67 | 19 | 7 | 5 | 2 | 100 |
| %age | 67% | 19% | 7% | 5% | 2% | |

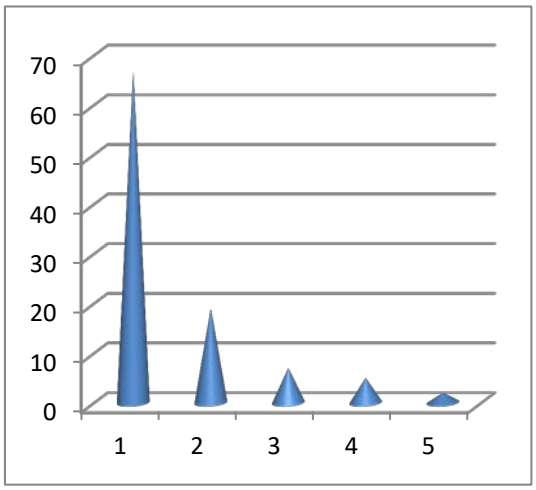


**Figure 3:** Design Phase Acceptance

**Table 7:** Development Phase Acceptance

| Development Phase Rule | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree | Total |
|---|---|---|---|---|---|---|
| Check List | 18 | 5 | 2 | 0 | 0 | 25 |
| Language | 20 | 2 | 2 | 1 | 0 | 25 |
| Defensible | 11 | 9 | 1 | 3 | 1 | 25 |
| Assessment | 16 | 1 | 2 | 2 | 4 | 25 |
| Pair Programming | 20 | 2 | 1 | 1 | 1 | 25 |
| Flexible | 23 | | 1 | 1 | 0 | 25 |
| Total | 108 | 19 | 9 | 8 | 6 | 150 |
| %age | 72% | 12.66% | 6% | 5.33% | 4% | |

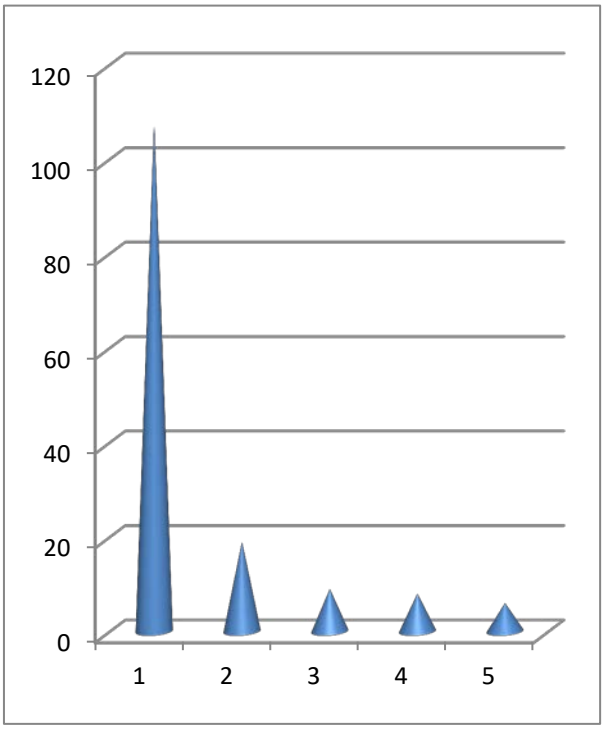


**Figure 4:** Development Phase Acceptance

**Table 8:** Testing Phase

| Rule | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree | Total |
|---|---|---|---|---|---|---|
| Testing Plan | 16 | 1 | 2 | 2 | 4 | 25 |
| Analysis Tool | 20 | 2 | 1 | 1 | 1 | 25 |
| Acceptance Level | 36 | 3 | 3 | 3 | 5 | 50 |
| Total | 72 | 6 | 6 | 6 | 10 | 100 |
| | 72% | 6% | 6% | 6% | 10% | |

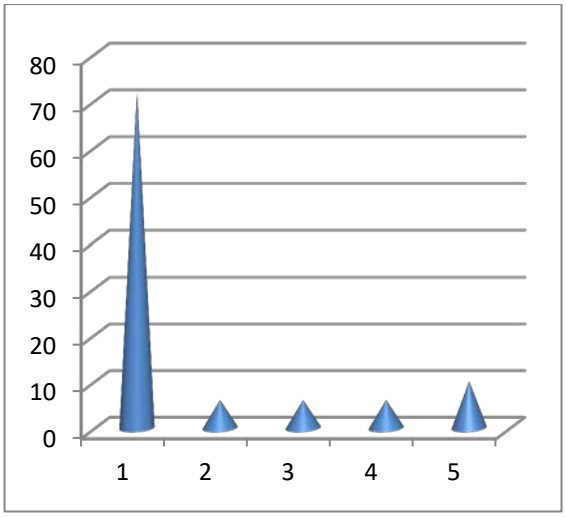


**Figure 5:** Testing phase Acceptance

**Table 9:** Overall SDLC Acceptance

| SDLC Phases | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |
|---|---|---|---|---|---|
| Requirement | 68.5% | 15.5% | 6% | 6% | 4% |
| Design | 67% | 19% | 7% | 5% | 2% |
| Development | 72% | 12.66% | 6% | 5.33% | 4% |
| Testing | 72% | 6% | 6% | 6% | 10% |
| Total | 69.87% | 13.29% | 6.25% | 5.58% | 5.00% |

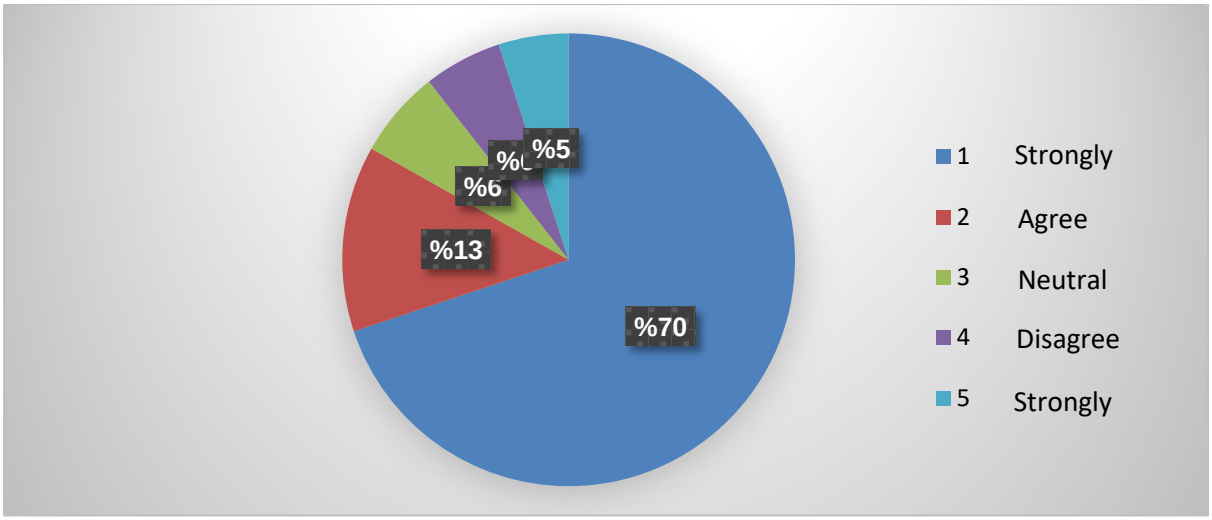


**Figure 6:** Overall SDLC Acceptance

By incorporation of security rules that are presented in this research (Table: 4) the imminent risk of a security breach in applications can be avoided to a greater extent. The rules of Requirement Phase include stakeholder identification, requirement elicitation, stakeholders satisfaction, removal of conflicts, compliance with international standards, threats identification, prevention, unambiguity etc. Design Phase of SDLC includes rules of following recommended development frameworks, security principles, external review, and access control. Development phase should comprise rules of checklist, language, defensible, assessment, pair programming, and flexibility. And Testing Phase of SDLC should embrace the testing plan, analysis tools, and acceptance level.

## 5. Conclusion

We cannot eliminate the importance of security in a successful software. Most of the times peoples implement after the development of software but it creates rigid boundaries and difficult to encourage security rules. This research gives us a platform to develop secure software by implementing these rules in respective phases of software development life cycle and this thing is validated by the research community and software engineers busy in different software houses to develop applications. In the future, we implement these rules in the development of different nature applications like web, desktop, cloud etc. and investigate the commonalities in all.

## References

[1] N. S. A. 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. 9, pp. 5333-5345, 2016.

[2] N. Haridas, "Software Engineering-Security as a Process in the SDLC," SANS Institute, p. 29, 2007.

[3] C. Banerjee and S. Pandey, "Software Security Rules, SDLC Perspective," arXiv preprint arXiv:0911.0494, 2009.

[4] A. Sharma and P. K. Misra, "Aspects of Enhancing Security in Software Development Life Cycle," Advances in Computational Sciences and Technology, vol. 10, pp. 203-210, 2017.

[5] A. Batcheller, S. C. Fowler, R. Cunningham, D. Doyle, T. Jaeger, and U. Lindqvist, "Building on the Success of Building Security In," IEEE Security & Privacy, vol. 15, pp. 85-87, 2017.

[6] N. Leicht, I. Blohm, and J. M. Leimeister, "Leveraging the Power of the Crowd for Software Testing," IEEE Software, vol. 34, pp. 62-69, 2017.

[7] C. Kumar and D. K. Yadav, "Software defects estimation using metrics of early phases of software development life cycle," International Journal of System Assurance Engineering and Management, vol. 8, pp. 2109-2117, 2017.

[8] S. A. Aljawarneh, A. Alawneh, and R. Jaradat, "Cloud security engineering: Early stages of SDLC," Future Generation Computer Systems, vol. 74, pp. 385-392, 2017.

[9] V. Tasril, M. B. Ginting, and A. P. U. S. Mardiana, "Threats of Computer System and its Prevention," International Journal of Scientific Research in Science and Technology, vol. 3, pp. 448-451, 2017.

[10] C. F.-G. R. Rios, and J. Lopez, "Modelling Privacy-Aware Trust Negotiations," Computers & Security, 2017.

[11] C. E. Landwehr and A. Valdes, "Building Code for Power System Software Security," Technical Report. IEEE Computer Society, Mar2017.

[12] O. Tripp and O. Weisman, "Identifying stored security vulnerabilities in computer software applications," ed: Google Patents, 2018.

[13] P. P. Choudhury, K. Dihidar, A. R. Khan, R. Verma, and P. Sarkar, "Software measurements and metrics: Role in effective software testing," Data, in Brief, vol. 3, pp. 593-596, 2017.

[14] B. C. f. P. S. S. Security. (2017). Available: https://smartgrid.ieee.org/images/files/pdf/building_code_for_power_system_software_security.pdf

[15] N. M. Mohammed, M. Niazi, M. Alshayeb, and S. Mahmood, "Exploring software security approaches in software development lifecycle: a systematic mapping study," Computer Standards & Interfaces, vol. 50, pp. 107-115, 2017.

[16] S. Rajesh and A. Chandrasekar, "Esteemed software patterns for the banking system," Cluster Computing, pp. 1-13, 2017.

[17] S. T. Siddiqui, "TSSR: a proposed tool for secure software requirement management," 2015.

[18] D. P. Gilliam, T. L. Wolfe, J. S. Sherif, and M. Bishop, "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, 2003, pp. 243-248.

[19] R. A. Majid, N. L. M. Noor, W. A. W. Adnan, and S. Mansor, "A survey on user involvement in software development life cycle from practitioner's perspectives," in Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on, 2010, pp. 240-243.

[20] G. McGraw, "Software Security," IEEE Security & Privacy, vol. 2, pp. 80-83, 2004.

[21] R. L. Krutz and R. D. Vines, Cloud security: A comprehensive guide to secure cloud computing: Wiley Publishing, 2010.

[22] G. Díaz and J. R. Bermejo, "Static analysis of source code security: Assessment of tools against SAMATE tests," Information and software technology, vol. 55, pp. 1462-1476, 2013.

[23] K. Sahu, R. Shree, and R. Kumar, "Risk management perspective in SDLC," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, 2014.

[24] B. Potter and G. McGraw, "Software security testing," IEEE Security & Privacy, vol. 2, pp. 81-85, 2004.

[25] P. Hope, G. McGraw, and A. I. Antón, "Misuse and abuse cases: Getting past the positive," IEEE Security & Privacy, vol. 2, pp. 90-92, 2004.

[26] N. Ayewah, D. Hovemeyer, J. D. Morgenthaler, J. Penix, and W. Pugh, "Using static analysis to find bugs," IEEE Software, vol. 25, 2008.

[27] S. Barnum and G. McGraw, "Knowledge for software security," IEEE Security & Privacy, vol. 3, pp. 74-78, 2005.

[28] N. S. A. Karim, T. Saba, and A. Albuolayan, "Analysis of software security model in the scenario of Software Development Life Cycle (SDLC)," Journal of Engineering Technology (ISSN: 0747-9964), vol. 6, pp. 304-316, 2017.

[29] J. Whitmore and W. Tobin, "Improving Attention to Security in Software Design with Analytics and Cognitive Techniques," in Cybersecurity Development (SecDev), 2017 IEEE, 2017, pp. 16-21.

[30] S. Barnum and M. Gegick, "Build security in– Design principles, 2005," ed.

[31] N. Davis, "Secure software development lifecycle processes: A technology scouting report," Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst2005.

[32] C. G. Cobb, "Fundamental Principles behind SDLC Models," Making Sense of Agile Project Management: Balancing Control and Agility, pp. 131-162, 2011.