



## Secure Software Development Process: A Survey

Mubashirah Majeed\*, S. M. K Quadri

Department of Computer Sciences, University of Kashmir,  
Srinagar, India

---

**Abstract**—Software security has become a great challenge as the rate of breaches is increasing. The main reason behind the security breaches of software systems is the lack of security consideration during the early development stages. There are two views in software development, Product view and process view. Former is concerned is what is to be developed and the latter is concerned with how it is to be developed. Security mainly comes under the process view of software development. Incorporation of security in the early stages of software development is a very young field and still the real development practices are lacking in addressing security issues early development phases. In this paper a thorough analysis of the current approaches towards secure software development have been presented. A survey has been carried out on the existing efforts in this field. For each of the Development stages the survey is based on the (i) Efforts made at each of the development stages (ii) Tools and techniques employed at each of the stage (iii) Level of applicability in real practices. The output of the survey is presented in graphical form for better insight into the current trend & efforts in secure software development processes.

**Keywords:** Security, Secure software Life cycle, Software systems, architectural, development

---

### I. INTRODUCTION

Information system encompasses many components like people, hardware, software, data, and networks. Each of these components must be secure enough for the smooth and reliable functioning of the information system. One of the most important components relative to the all other components of information systems is the software systems. Incorporating the security in software product or system is a long standing challenge for software developers. Developing secure software is not an advantage but has become a necessity for the software organization. In Today's networked environment, the software is becoming more vulnerable to both the deliberate and accidental malicious intents. The main reason behind the security holes in the software is due to the negligence of addressing security issues in the software development process. Security is always treated as an afterthought in the software development process, and dealt with after the system development stages by providing the required preventive measures. Security needs to be considered from the very beginning of the software development life cycle [1], [2]. Such secure software development process should start with the security requirements, known as NFR (Non Functional Requirements) along with the procedural software requirements, followed by the secure design and implementation. This paper presents a detailed analysis and the survey of the current approaches and efforts that have been made towards secure software development lifecycle (SSDLC).

### II. SOFTWARE SECURITY

Software security is to engineer software in such a way that the desired application functions uninterrupted and is able to properly handle the security threats during malicious attacks. Security ensures that application works in a desired manner and to provide defence against security threats. In common practice, security is unnoticed in early phases of software development life cycle (SDLC). A good software engineering approach is to think about security right from beginning of SDLC. Inadequate practice of software development can lead to insecure software [3].

### III. CURRENT APPROACHES AND SCENARIOS

In an effort to produce a secure software product, there are mainly three approaches that are followed in real practices [4].

#### A. Penetrate and patch:

This approach aims at the application of patches after the detection of vulnerabilities in the system. There are two drawbacks to this approach, first the application of patches results in further vulnerabilities exploitation once the attacker came to know about the patch, secondly it is believed that fixing the bug after the release of a software is almost 100 times more expensive than putting the efforts in the system development stages [5].

#### B. Secure operational environment:

This approach focus on employing the protection mechanisms in the operational environment like firewalls, intrusion detection systems etc.

### C. Secure software engineering:

This approach is actually the most structured way of developing secure software by taking the security right from the scratch i.e. from the requirement to the design, implementation and testing stages of software development.

There is a difference between the software security and application security. Software security is concerned with addressing the security in the development phases of the software whereas application security is concerned with protecting the software after the development phases by mean of various protection mechanisms like intrusion detection , firewalls etc. Secure software development focus on the security issues right from the beginning from requirement, design and implementation phases of SDLC. Many ideas of secure software development (SSD) methods have been proposed [6]. However the process of software development still follows the conventional SDLC models and process. In their study [6] have summarized the various secure software development lifecycle (SSDLC) processes and also provided the detailed comparisons of these processes based on the identified characteristics.

Typically SSDLC is comprised of following phases [7]

1. Software Security Requirements
2. Secure Software Architecture & Design
3. Secure Software Implementation and Coding
4. Security Assurance

Figure 1 depicts the stages for secure software development process with the core activities to be carried out at each of the stage. These stages only tell what is to be achieved not the how it can be achieved. The ingredients of each of the above mentioned stage vary from one approach to other. The difficulty behind the specification of operations that can be performed in each of the stages of the lifecycle in order to develop the secure software is due to multifaceted nature of the software security [8]. It is a complex problem to identify which factors constitute to the security of the system, also the software system have different environment factors under which they operate. Such dependencies create an uncertainty about the fact that which technique is the best.

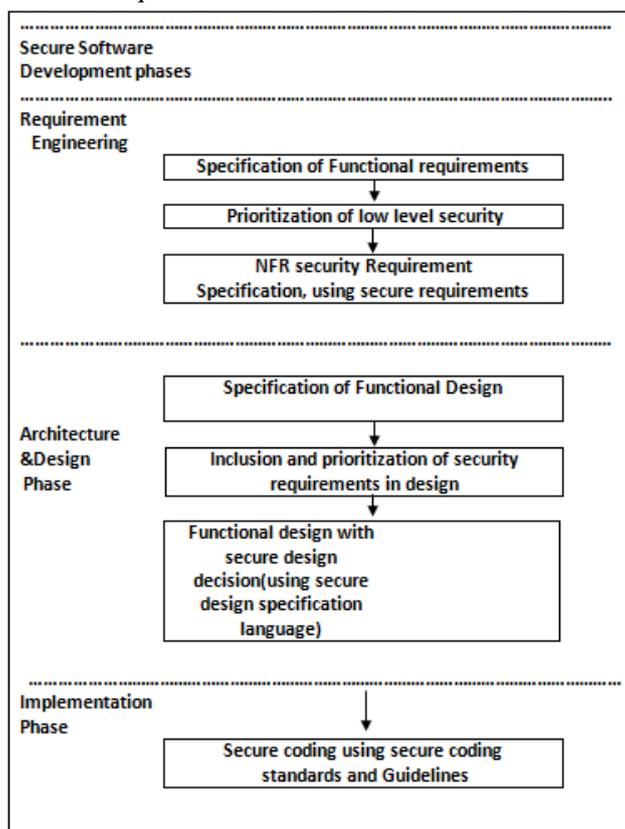


Figure 1: Secure software development Lifecycle

## IV. SOFTWARE SECURITY REQUIREMENTS

It is believed and estimated that an error introduced in the requirement phase can cost up to 100times more to correct in the further development stages [9]. A security requirement act as the manifestation of a high-level organizational security policy in the detailed requirements of a specific system. The most current software requirement specifications fall within the following three categories [10]

- (i) Totally silent regarding security.
- (ii) Merely specify vague security goals or
- (iii) Specify commonly used security mechanisms as architectural constraints.

In the first case security is not considered in the software development at all, in the second case the specified security requirements are unstructured and very hard to be evaluated. Third case binds architectural decision too early in the at the requirement phase, which results in inappropriate security mechanism. Software security requirements come under the NFRs (Non-functional Requirements) which are quite different from the functional requirements. NFR describe how the software will perform rather than what it performs. NFRs are subjective in nature like maintainability, performance, reliability security etc. From the literature survey, still there exist no concrete, well established and unambiguous non-functional security requirements specification criteria, and also the lack of metrics for specifying objectives tolerance of these requirements, which poses the difficulties in evaluating the system against the specified requirements. If software is developed without consideration of the security requirements, the more work gets shifted to later design phase of the software development to address and remove such security holes incurred during the requirement phase. This can lead to security flaws in the design phase, because the primary focus of the developers remains on the functional parts rather than the non-functional aspect of the system. Exploitation of security flaws in such software occurs due to the circumvention of security mechanism rather than breaking such mechanisms [11].

## **V. SECURITY REQUIREMENTS ENGINEERING**

Requirement engineering is the most important and fundamental step towards the quality software development. Security requirements are not precise enough and need to be more explicit about who can do what and when [12]. Generally software requirements engineering can be split into requirement development and requirement management [13]. The requirement development involves the activities like gathering, evaluating and documenting the requirements and requirement management entails establishing and maintaining an agreement with customer on the requirements [14].

## **VI. SECURITY REQUIREMENTS SPECIFICATION LANGUAGES**

There are various security specification languages such as UMLsec [15], secureUML [16], SecureTropos [17], Misuse Case [18], AbuseCase [19], UMLintr [20] and AsmlSec [21].

## **VII. SECURE SOFTWARE ARCHITECTURE & DESIGN**

Design is the blueprint of the overall structure of the software, it address the definition of the overall structure of the software. It is the design phase where the decision is to be taken to fulfill the specified requirements. The incorporation of security into the software design process is influenced by both the software design concepts and security methods. Defining the overall structure of the software from the security perspective entails identifying those components whose correct functioning is essential to security and identifying design techniques that will assure its security. Architecture & design specifies two aspects of the software: Static structure and Dynamic Behavior. According to [7]in Secure Software Development process(SSDP), architecture & design phase involves following steps.

1. Detailed functional design including the security mechanisms should be specified by following the secure design guidelines and patterns. The security specification languages like UMLsec can be utilized at the end.
2. The design should be inspected repeatedly for the identification and removal of errors.
3. The threat modeling which is normally carried out in the requirement phase of secure software development should also be enhanced and applied in architecture and design phase.
4. Based on the identified threats, the risk analysis should be performed to calculate the potential damage of each of the identified threat.
5. The prioritization of secure design decision should be carried out to remove threats based on cost-benefit analysis.
6. Previous specified secure design decision should be identified. The Security vulnerabilities in the existing similar software can also be used as checklist.
7. A threshold of acceptable security needs to be defined. Security index of the design must be within the threshold.
8. If the calculated security index not satisfies the threshold level, then secure design to remove the errors should be specified. The secure software design is still in the early stages; in practice secure software design guidelines are widely used. These secure design guidelines are the suggestion and the principles based on the experience the developers have gained while solving software security problems repeatedly. In [22] proposed three software security design principles that are intended to associate software and security design concepts. The proposed principles are:

*Principle 1:* The software design has multiple iterative phases and the security features should be incorporated and adjusted during each of those phases.

This author [22] suggests that the principle 1. Can be achieved by utilizing the UML(Unified Modeling language). From the UML use case diagrams representing the requirements, the UML class diagrams at the higher level of abstraction with only attributes and methods signature should be specified. The designer needs to return to the UML use case and associate the use case with the classes as required. This may also result in the design of new classes between actors and use case. With the increasing maturity of the design, the use case of requirement phase and the associated design phase

classes can be combined together into the sequence diagrams. As the process of design progress further, other UML diagrams such as collaboration, object, state, activity diagrams etc. may be supplied. With each sub phase of the design phase the security level of the design gets improved repeatedly

*Principle 2:* Security assurance is relative to the phase of software design and the chosen Security Assurance Rules (SARs).

This principle stipulates that the security assurance be evaluated against the respective software design phase to constantly enforce Software Assurance Rules (SARs). The author in [22] suggests that the principle 2 can be achieved by utilizing the SARs (Software Assurance Rules). These SARs are derived from the higher level security policies. The idea is, as the design context changes from one to the next in UML, different SARs are to be utilized to enforce the security features for that sub-phase of design.

*Principle 3:* The security incorporating process should neither counter the intuition nor decrease the productivity of the software designer. In literature various principles for secure software design have been proposed. To start with, first set of guidelines proposed in [23]. These guidelines have elaborated by providing the examples in [24]. In their study, [25] have proposed secure sets of secure design guidelines. In his study [26] have analyzed and refined the guidelines proposed in the earlier studies as result of which a consolidated set of secure software design guidelines by adding and modifying the adequate guidelines from the existing sources and also some newly proposed have been reported.

## VIII. SECURE DESIGN SPECIFICATION LANGUAGE

The UML diagrams are heavily used in the design phase of the software development. UML is equipped with varieties of diagrams to specify the functional and behavioral aspects of the software. UML is one of the dominant modeling languages used in the software design for specification, visualization, construction of software artifacts. The extended UML such as UMLsec [15], secureUML [16], SecureTropos [17], MisuseCase [18], AbuseCase[19], UMLintr[20], AsmlSec[21] discussed above are also used for the specification of the secure software design. According to[7], there are two major concerns that should be considered in selecting a secure design language. These concerns are the varieties of diagrams available to represent the design from various perspectives with certain level of abstraction and the availability of tools. Varieties of tools provided by UMLsec can be utilized for secure software design. SecureUML on the other hand can also be used in secure software design; however it is limited to representing only role-based access control notions in a UML class diagram. SecureTropos [27] proposes varieties of use Agent diagrams such as agent interaction diagram, plan diagrams which are similar to the UML activity diagrams and sequence diagrams. With the help of such secure design languages the security of the software can be specified explicitly in both the requirement and design phase in order to promote secure software development.

## IX. SECURE SOFTWARE IMPLEMENTATION (CODING)

For the secure software development process the security needs to be taken into consideration from the requirement to design followed by the implementation (coding). Secure software implementation (coding) involves the process and application of secure coding standards and guidelines, application of security testing tools including "fuzzing", static-analysis code scanning tools, and conducting code reviews in order to eliminate the vulnerabilities in the code. Various studies showed that much of the vulnerabilities creep into the system through the poor coding techniques. There is no widely accepted standard(s) for secure programming. Programmers today must instead rely on techniques, examples, guidelines, rigorous testing and an awareness of risks to be avoided, in order to write reasonable secure code [28]. Many secure coding guidelines have been proposed in the literature but are very difficult to apply in practice. Some of the guidelines spread over the hundreds of rules, some aim to prevent the very specific type or errors. Also the choice of the language used is a key consideration in secure coding process which is beyond the scope of current study. Various secure coding guidelines tend to have little effect on the programmers when they write code and their benefit is very small in practice. His study [29] have analyzed the problem with the existing secure coding guidelines and principle as:

- The worst aspect of various secure coding guidelines is that they not allow for comprehensive tool-based compliance check. Tool based check is required, since it is infeasible to check the hundreds and thousands lines of code manually.
- The set of the rules have to be small and clear enough, so that it can be easily understood and applied by the programmers. Regarding this aspect the author argued that, it will be beneficial significantly by restricting the number of rules below or up to ten in numbers.

In the same work [28] proposed a set of rules for safety critical coding. This set comprises of ten rules along with the brief rationale for the inclusion of the each of the rule in real practices.

## X. SECURITY IN SOFTWARE DEVELOPMENT: A SURVEY

In the current networked environment most of the organizations rely on the software system for the storage and processing of data. Any deliberate or unintentional bug if get exploited can lead to a serious problem with respect to the confidentiality of the overall system and organizations. According to CERT [29] there has been considerable increase in the vulnerabilities in the last few years. Below diagram depicts the rate of the increase in vulnerabilities detection [30].

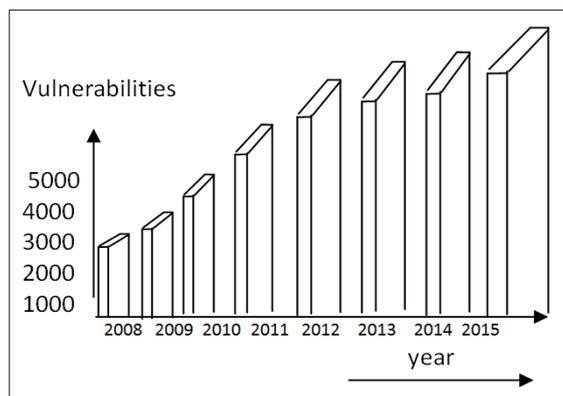


Figure 2: vulnerability reports.

As the rate of vulnerabilities increased, security has been taken as a serious challenge over the last few years [31]. In this section a detailed survey of some recent efforts towards secure software development process has been presented. In our benchmark, generalized secure software development lifecycle has been selected that covers the key stages of the development process. The parameters for the current benchmark are:

- I. Activities carried out in each of the above mentioned stage.
- II. Tools and techniques that have been employed at each of the stage.
- III. Level of Applicability in real practices.

The first parameter just presents the conceptual aspect of the particular selected effort in an abstract manner. The second parameter specifies the tools and the techniques adopted to achieve the activities carried out at first stage. The third parameter, level of applicability is dependent upon many factors under different conditions. In our study we have quantified the level of applicability into three categories as low, moderate and high. From this the level of applicability is further based on three parameters, (i). Knowledge of security required in implementation (ii). Availability of automated tools (iii) If automated tools are not available the amount of manual work to be done to achieve the desired goal.

Below Table 1 will present the selected efforts towards secure software development process in an abstract manner. We have categorized the efforts into four basic class's i.e. analysis, design, coding & Testing. Further each of the above mentioned parameters have been looked upon and analyzed& specified. The entry (x) in the table 1 specifies that the efforts have not been made under that class.

## XI. RESULTS AND DISCUSSION

As presented in table 1 the efforts made towards the security in software development process are sporadic in nature. It is very hard to find out any linear path in the efforts made. The reason behind the discontinuity and such large spectrum of different and divergent efforts is because of the nature of the problem. The security of a software system depends upon the various independent factors such as size, nature of the operational environment etc. and also for each organization there is a different set of security preferences. Still there is great need to streamline the efforts towards secure software development process. Further majority of the efforts made have very less feasibility & significance in the real practices.

Below figure 3 to 7 depicts the result of this survey in a graphical manner. Figure 3 depicts the overall frequency of efforts made at the identified stages (analysis, design, coding, testing) of secure software development process. The level of applicability at each of the stage have been calculated and presented in the below figure 4, 5, 6 and 7 respectively. From this survey, as depicted in the below figure 3, the highest level of applicability with respect to our identified parameters is of the analysis and the testing phase of the software development process. The Design and the coding phase have the lowest applicability level with respect to our identified parameters. Among all these, the efforts made at the design phase having the least applicability.

## XII. CONCLUSION

Secure software development demands a great deal of efforts because of the multistage phases of the software development life cycle. The secure software development process is twofold in nature. On one hand we still lack the secure software development mechanism that can be widely adopted and on the other hand a mechanism is needed to evaluate the effectiveness of the applied mechanism. The difficulty behind the immaturity of secure software development process is due to the fact that multiple external factors and the variation in the actual operating environment. From our survey we found that most of the efforts towards secure software development are made at the design and testing phases, which is very much on the track since design act as a blueprint of the entire system. Also the survey revealed that a great percentage of efforts made are not applicable in the real practices. Our future efforts will be the identification of key factors and parameters responsible for the security of overall system at each of the above mentioned phase (Fig.1) of secure software development life cycle. Based on the indicators of the current survey and the identified parameters we will propose a hybrid secure software development cycle that can be generic in nature and applicable in any environment.

Table 1. Data collection and categorization of efforts towards secure software development

SDLC PHASES AU	Security Analysis			Secure Design			Secure Coding			Security Testing		
Authors	Efforts	Tools & Techniques	Level of Applicability	Efforts	Tools & Techniques	Level of Applicability	Efforts	Tools & Techniques	Level of Applicability	Efforts	Tools & Techniques	Level of Applicability
Dianxiang, et al[32]	NA						NA			Automated generation of security tests for Vulnerability detection by using threat models.	Threat modeling.	High
Curtis Steward, et al [33]	Proposed an assurance based development model (ABD) for secure s/w development	Static code analysis tools (STAs), Complexity analysis, and changing development models.	Low	NA			NA			NA		
John Diannant, et al [34]	Presents a concept for focusing on vulnerabilities earliest in the development process.	Requirement gap analysis.	High	Concept for mitigation of risks in the design phase.	Architectural threat analysis.	Moderate	Concept for dealing with security in the coding phase.	Static application security testing.	Moderate	Concept for dealing with vulnerabilities in testing phase.	Dynamic application security testing	Low
Shafiq Hussain, et al [35]	NA			Concept for Threat modeling to make s/w design more secure.	STRIDE/CORAS, Abuse Stories, Attack trees, Fuzzy logic.	Low	NA			NA		
A.Adebiyi, et al[36]	NA			Method for evaluative security in s/w design	neural networks	High	NA			NA		
Steve Lipner,[37]	Theoretical view	NA	Low	Hypothetical view	Defined some guidelines	Low	Theoretical view	Idea of threat modeling	Low	Theoretical view	Static analysis tools, fuzz testing, code review.	Low
Per Håkon Meland, et al [38]	NA			Framework for secure design.	Guidelines and principles.	Low	NA			NA		
Axelle apville, et al [39]	Methodology for considering security at each level of development.	Threat modeling and risk evolution.	Moderate	concept	Security pattern and umlsec.	Moderate	Concept.	Cryptography.	Moderate	NA		
Malik Inran Daud, [31]	Uses XP techniques to recheck requirements at every stage of development	Security functional requirements, security non-functional requirements, derived requirements.	Moderate	Finds out the entry points and exit points in the system.	Threat modeling.	Moderate	Identified vulnerabilities and their countermeasures.	Taking care of the identified vulnerability and countermeasures in the development process.	High	To check whether the software has met the security requirements.	Functional testing, risk based testing, penetration testing and fuzz testing.	Low
Russell L. Jones, et al [40]	Proper Requirement gathering.	Conduct interviews and workshops.	Low	Identifies various risks and establishes various risk mitigation strategy	Threat modeling, risk assessment and asset identification and valuation, risk mitigation, security review.	Low	Provides various security guidelines and rules for secure coding.	Set of principles and guidelines.	low	Concept for performing various testing techniques for secure s/w development.	Unit testing, quality assurance testing, penetration testing.	Low
Brad Arkin, et al [41]	NA			NA			NA			Concept for application of penetration testing in the secure s/w development.	Static analysis tools, dynamic analysis tools, fuzzing.	Low
GARY MCCGRW, [42]	Theoretical concept of requirement gathering.	Abuse cases	Moderate	Concept on secure design principles and guidelines.	Risk analysis, external review.	Low	Concept only	Static analysis tool.	Low	concept	Penetration testing.	Low
Zeineb Zhioua, et al [43]	NA			NA			NA			Explains various static code analysis techniques for secure s/w development lifecycle.	Model checking, control flow analysis, data flow analysis, information flow analysis	Moderate
A. S. Sodiya, et al [44]	Proposes a model that integrates security engineering with s/w development process.	Security training, threat modeling.	Low	Concept	Security specification(SS),review SS.	Low	NA			concept	Penetration testing	Low
Gary McGraw, [45]	NA			NA			Conceptual view on code security.	Code review tools	Low	NA		
Takao Okubo, et al [46].	NA			NA			Detailed concept for preventing various injection attacks like SQL injection and	Coding convention and proposed security framework.	High	NA		
Humming Yu, et al [47]	NA			NA			Cross site scripting (XSS) in web applications.	Some secure programming guidelines.	Low	NA		
							Describes a course module for improving the programming practices for writing secure code.					

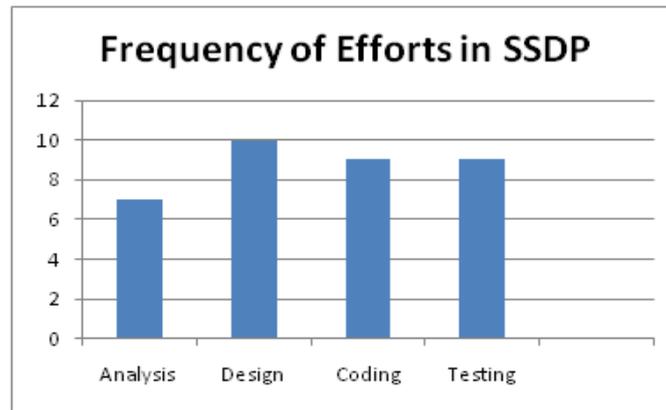


Figure 3. Level of Efforts made towards secure software development at different identified stages

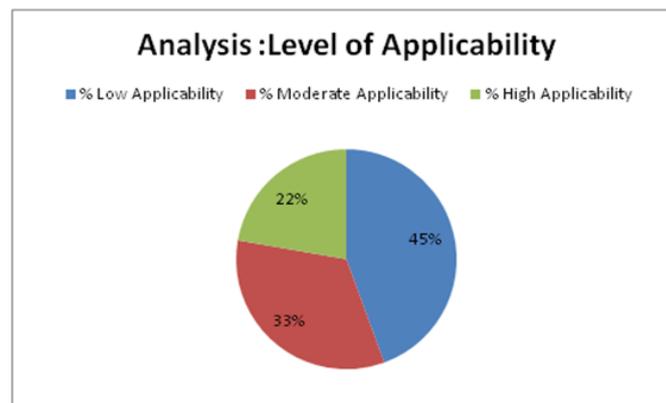


Figure 4. Applicability level of efforts made at analysis phase of secure software development.

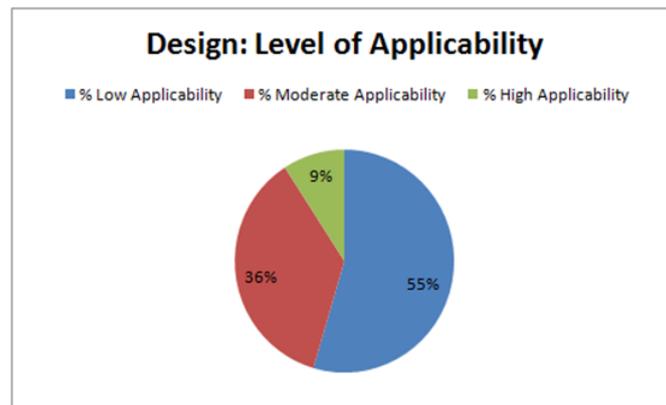


Figure 5. Applicability level of efforts made at Design phase of secure software development.

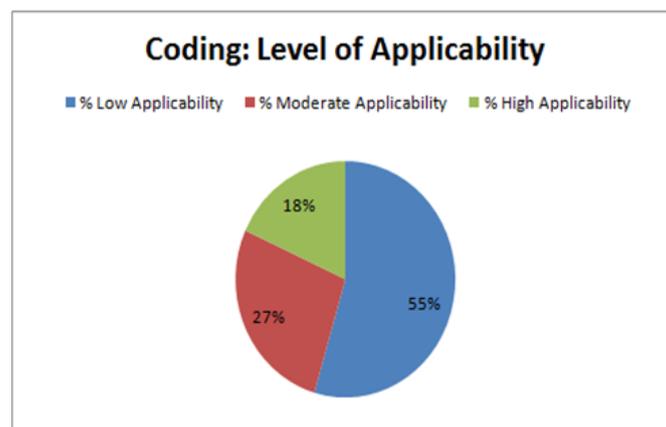


Figure 6. Applicability level of efforts made at Coding phase of secure software development.

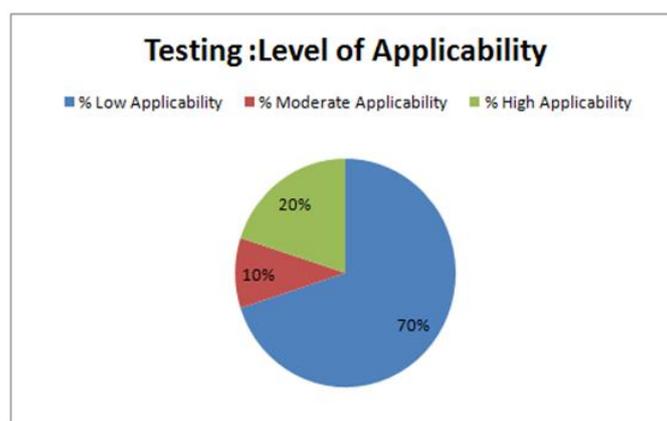


Figure 7. Applicability level of efforts made at testing phase of secure software development.

## REFERENCES

- [1] WANG, HUIQING, AND CHEN WANG. Taxonomy of security considerations and software quality. *Communications of the ACM* 46.6 (2003): 75-78.
- [2] DEVANBU, PREMKUMAR T., AND STUART STUBBLEBINE. Software engineering for security: a roadmap. *Proceedings of the conference on the future of Software engineering*. ACM, 2000.
- [3] C. Mann, "Why Software is so Bad" *Technology Review* (July/August 2002)
- [4] SHIRAZI H. M., A New Model for Secure Software Development. *International Journal of Intelligent Information Technology Application*, 2009, 2(3):136-143
- [5] BOEHM, BARRY W. Industrial software metrics top 10 list. *IEEE software* 4.5 (1987): 84-85.
- [6] KHAN, MUHAMMAD UMAIR AHMED, AND MOHAMMAD ZULKERNINE. A Survey on Requirements and Design Methods for Secure Software Development. No. 2009- 562. Technical Report, 2009.
- [7] KHAN, MUHAMMAD UMAIR AHMED, AND MOHAMMAD ZULKERNINE. Activity and Artifact Views of a Secure Software Development Process. *Computational Science and Engineering*, 2009. CSE'09. International Conference on. Vol. 3. IEEE. 2009.
- [8] Mir, Irshad Ahmad, and S. M. K. Quadri. "Analysis and evaluating security of component-based software development: A security metrics framework." *International Journal of Computer Network and Information Security (IJCNIS)* 4.11 (2012): 21.
- [9] BOEHM, BARRY W., AND PHILIP N. PAPACCIO. Understanding and controlling software costs. *Software Engineering*, *IEEE Transactions on* 14.10 (1988): 1462-1477.
- [10] FIRESMITH, DONALD. Specifying reusable security requirements. *Journal of Object Technology* 3.1 (2004): 61-75.
- [11] JÜRJENS, JAN. Using UMLsec and goal trees for secure systems development. "Proceedings of the 2002 ACM symposium on applied computing. ACM, 2002.
- [12] MCGRAW, GARY, AND BRUCE POTTER. Software security testing. *IEEE Security and Privacy* 2.5 (2004): 81-85.
- [13] WIEGERS, KARL E. *Software requirements*. Microsoft press, 2009.
- [14] PAULK, M. C., et al. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison Wesley. 1995.
- [15] JÜRJENS, JAN. *Secure systems development with UML*. Springer, 2004.
- [16] LODDERSTEDT, TORSTEN, DAVID BASIN, AND JÜRGEN DOSER. SecureUML: A UML- based modeling language for model-driven security. «UML» 2002—The Unified Modeling Language (2002): 426-441.
- [17] FIRESMITH, DONALD G. Security use cases. *Journal of object technology* 2.3 (2003).
- [18] SINDRE, GUTTORM, AND ANDREAS L. OPDAHL. Eliciting security requirements with misuse cases. *Requirements Engineering* 10.1 (2005): 34-44.
- [19] MCDERMOTT, JOHN, AND CHRIS FOX. Using abuse case models for security requirements analysis. *Computer Security Applications Conference, 1999.(ACSAC'99) Proceedings. 15th Annual. IEEE, 1999.*
- [20] HUSSEIN, MOHAMMED, AND MOHAMMAD ZULKERNINE. UMLintr: a UML profile for specifying intrusions." *Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on. IEEE, 2006.*
- [21] RAIHAN, MOHAMMAD, AND MOHAMMAD ZULKERNINE. AsmLSec: an extension of abstract state machine language for attack scenario specification. *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on. IEEE, 2007.*
- [22] DOAN, THUONG, ET AL. "MAC AND UML FOR SECURE SOFTWARE DESIGN. *Workshop on Formal Methods in Security Engineering: Proceedings of the 2004 ACM workshop on Formal methods in security engineering. Vol. 29. No. 29. 2004.*

- [23] SALTZER, JEROME H., AND MICHAEL D. SCHROEDER. The protection of information in computer systems." Proceedings of the IEEE 63.9 (1975): 1278-1308.
- [24] BISHOP, MATT. Introduction to computer security. Addison-Wesley Professional, 2004.
- [25] HOWARD, MICHAEL, AND DAVID LEBLANC. Writing secure code. Microsoft press, 2009.
- [26] PEINE, HOLGER. Rules of thumb for developing secure software: Analyzing and consolidating two proposed sets of rules. Availability, Reliability and Security, 2008. ARES 08. Third International Conference on. IEEE, 2008.
- [27] BAUER, BERNHARD, JÖRG P. MÜLLER, AND JAMES ODELL. Agent UML: A formalism for specifying multiagent interaction. Agent-oriented software engineering. Vol. 1957. Springer, Berlin, 2001.
- [28] GRAFF, MARK, AND KENNETH VAN WYK. Secure coding: principles and practices. O'Reilly Media, Incorporated, 2003.
- [29] HOLZMANN, GERARD J. The power of 10: rules for developing safety-critical code."Computer 39.6 (2006): 95-99.
- [30] Carnegie Mellon university, copyright © 1995-2009 [modified: February 12, 2009], cert, <http://www.cert.org/stats/>
- [31] 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).
- [32] Xu, D., Tu, M., Sanford, M., Thomas, L., Woodraska, D., & Xu, W. (2012). Automated security test generation with formal threat models. Dependable and Secure Computing, IEEE Transactions on, 9(4), 526-540.
- [33] Steward Jr, C., Wahsheh, L., Ahmad, A., Graham, J. M., Hinds, C. V., Williams, A. T., & DeLoatch, S. J. (2012, April). Software security: The dangerous afterthought. In Information Technology: New Generations (ITNG), 2012 Ninth International Conference on (pp. 815-818). IEEE.
- [34] Diamant, J. (2011). Resilient security architecture: a complementary approach to reducing vulnerabilities. Security & Privacy, IEEE, 9(4), 80-84.
- [35] Hussain, S., Kamal, A., Ahmad, S., Rasool, G., & Iqbal, S. (2014). Threat Modelling Methodologies: A Survey. Sci. Int.(Lahore), 26(4), 1607-1609.
- [36] Adebisi, A., Arreyemi, J., & Imafidon, C. (2012, March). Applicability of neural networks to software security. In Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on (pp. 19-24). IEEE.
- [37] Lipner, S. (2004, December). The trustworthy computing security development lifecycle. In Computer Security Applications Conference, 2004. 20th Annual (pp. 2-13). IEEE.
- [38] Per, H., & Jensen, J. (2008, March). Secure software design in practice. In The Third International Conference on Availability, Reliability and Security (pp. 1164-1171). IEEE.
- [39] Apvrille, A., & Pourzandi, M. (2005). Secure software development by example. IEEE Security & Privacy, (4), 10-17.
- [40] Jones, R. L., & Rastogi, A. (2004). Secure coding: building security into the software development life cycle. Information Systems Security, 13(5), 29-39.
- [41] Arkin, B., Stender, S., & McGraw, G. (2005). Software penetration testing. IEEE Security & Privacy, (1), 84-87.
- [42] McGraw, G. (2004). Software security. Security & Privacy, IEEE, 2(2), 80-83.
- [43] Zhioua, Z., Short, S., & Roudier, Y. (2014, July). Static Code Analysis for Software Security Verification: Problems and Approaches. In Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International (pp. 102-109). IEEE.
- [44] Sodiya, A. S., Onashoga, S. A., & Ajayi, O. B. (2006). Towards building secure software systems. Issues in Informing Science and Information Technology, 3, 635-646.
- [45] McGraw, G. (2008). Automated code review tools for security. Computer, (12), 108-111.
- [46] Okubo, T., & Tanaka, H. (2007, April). Secure software development through coding conventions and frameworks. In Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on (pp. 1042-1051). IEEE.
- [47] Yu, H., Jones, N., Bullock, G., & Yuan, X. Y. (2011, October). Teaching secure software engineering: Writing secure code. In Software Engineering Conference in Russia (CEE-SECR), 2011 7th Central and Eastern European (pp. 1-5). IEEE