Secure Software Development a Survey

Sunil Patel, Deepak Kulhare, Arif Khan

Abstract— Technology and its applications are raises day by day fashion, in our daily life we are various times interacted with different kinds of computer and its application that shows effects of technology in our daily life. To design and deploy an application that helps us on different utilities are made possible using the software engineering and its approaches. In this paper we provide the different aspects and issues on the traditional software development methodology, and discuss the proposed solution in the direction of optimize the approach to find better solutions with less effort and time. Additionally we focus mainly on the vulnerabilities in software engineering at the time of development and their solution. After all we propose a new way for scan and trace the vulnerabilities in software application development.

Index Terms — Vulnerabilities, software development, security, processes.

I. INTRODUCTION

Secure Software Development covers activities which lead to the development of quality software from a security point of view. Software development is a layered architecture of different activities. These activities are accepts intermediate results and through these results new results are generated. If any step is week then effect of this are reflect on complete system.

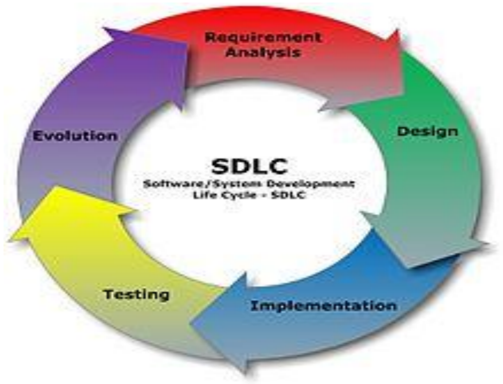

Fig 1 Software development life cycle

To understand the complete development life cycle, we start with the requirement analysis. That is a phase where developer team understand the requirement of end client, if any problem in the requirement then it is cleared by the developer because of problem in understanding leads a gap on the development.

Once the requirement is gathered required to develop a plan by which all the team members starts implementation of the system. That planning phase is called as design, here all the desired work is planed using the paper and pencil.

Manuscript received on April, 2013.

Sunil Patel, M.Tech Scholar, Computer Science & Engineering, CIIT Indore, Indore, India.

Deepak Kulhare, Associate Professor,Computer Science & Engineering, CIIT Indore, Indore, India.

Arif Khan, Assistant Professor Computer Science & Engineering, CIIT Indore, Indore, India.

Designed product is now read to become live, that is derived in Implementation phase. To maintain or deliver quality software testing is performed and finally after complete system is deployed over client end. Here we can see all the software development process activities are dependent over each other. Thus care and precaution are required for deliver a quality product.

To make more effective and efficient application development various process models are available where the developer team can manage their changes and other requirements.

In this section of our paper we provide the general introduction of SDLC and their dependency over each and every layer. In the next section we provide the advantages and issues of the different process model.

II. BACKGROUND

In this section of the work we provide the different process models introductions and their relevant advantages and disadvantages, additionally here we provide the effort and technologies made in previous research work by the our authors and researchers.

A. Waterfall

The waterfall development model originates in the manufacturing and construction industries; highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development. The main advantages to use this model are as follows:

- Simple goal.
- Simple to understand and use.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.
- Easy to manage. Each phase has specific deliverable and a review.
- Works well for projects where requirements are well understood.
- Works well when quality is more important than cost/schedule.
- Customers/End users already know about it.

As they have some advantages the model having some disadvantages too.\

- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- No working software is produced until late in the life cycle.
- Risk and uncertainty is high with this process model.
- Adjusting scope during the life cycle can end a project
- Not suitable for complex projects

- Not suitable for projects of long duration because in long running projects requirements are likely to change.
- Integration is done as a "big-bang" at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.
- Users can only judge quality at the end.
- Attempt to go back 2 or more phases is very costly.
- Percentage completion of functionality cannot be determined in mid of the project because Every Functionality is undergoing some phase.
- Very risky, since one process cannot start before finishing the other.

B. Incremental

Iterative and Incremental development is any combination of both iterative design or iterative method and incremental build model for development. The combination is of long standing and has been widely suggested for large development efforts. Some key benefits are

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration and Each Iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment operational product is delivered.
- Issues, challenges & risks identified from each increment can be Utilized/Applied to the next increment.

Disadvantages are:

- More resources may be required.
- Although cost of change is lesser but it is not very suitable for changing requirements.
- More management attention is required.
- Each phase of iteration is rigid with no overlaps.
- System architecture or design issues may arise because not all requirements are gathered. Up front for the entire life cycle.
- Does not allow iterations within an increment.
- Defining increments may require definition of the complete system.

And about the entire model are having some advantages and disadvantages, between these issues vulnerabilities is a more important issue in software engineering. In the next section we provide the previous researches and approaches that indicate the issues and aspects of the software development.

III. WORK DONE SO FAR

In this section we provide the different research conclusions and our study around the domain of interest. Vulnerabilities, if not uncovered and mitigated during software development, can incur huge cost in terms of time, money and efforts after implementation. Integrating security within the development life cycle has been proven to be the most effective way to develop secure software. To the aim, this paper [1] author proposes a framework to identify, analyze and mitigate vulnerabilities during the development life cycle.

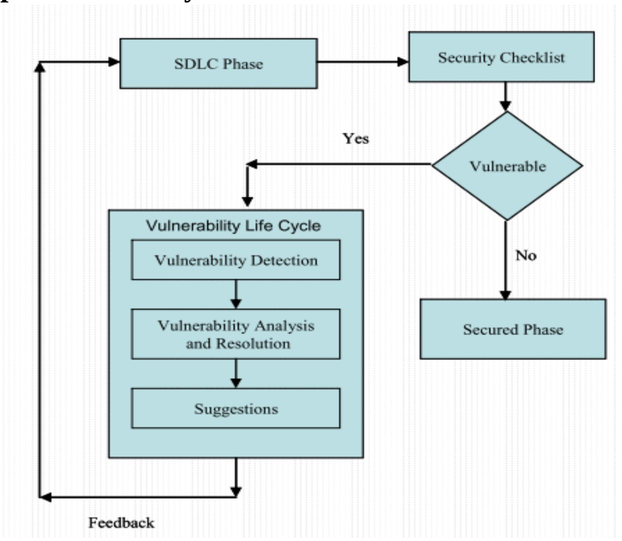

Fig 2 Software Vulnerability Detection & Analysis Framework

Security checklists have been proposed at various phases of software development life cycle to examine its vulnerability. The overall concept is based on the fig 2. The framework suggests technique for developing secure software. Security checklist is generated and used to verify whether security prerequisite has been fulfilled or not for each phase of SDLC. Vulnerability detection phase analyzes the vulnerable input at each phase and report the vulnerabilities so that through next phases of vulnerability life cycle, the vulnerable input can be modified to secure output, which is send back to the software life cycle in the form of feedback.

In [2] author provides Security considerations during each phase of a generic development life cycle. The generic phases used for this document are Planning, Analysis, Design, Implementation, and Support. Before expounding on the security considerations encapsulated in each phase, each major section of this document briefly discusses the spirit of the life cycle's phase, highlights the responsibilities of a security analyst during that phase, and often compares the similarities and differences between the developer and the security analyst's roles. In the end, a broad group of security topics are addressed. These include the differences between program and issue-specific policies, CIA (confidentiality, integrity, and availability) and Risk Assessment, different levels of Security Strategies, levels of an Application Security Management Plan and how to manage vulnerability assessments in application testing and other best practices.

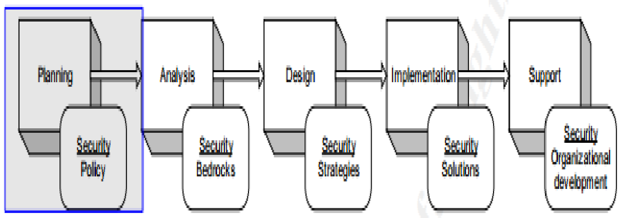

Fig 3 Baking Security in SDLC

This composite demonstrates a repeatable process where enterprise security policies allow business requirements to be met through the execution of appropriate security strategies and solutions, which contains the individual security policies which are required to apply individually.

Keeping in view the daily increase in software security threats, developing secure software has become a necessity and challenge. Early detection of vulnerabilities [3] in software while developing it and countering them in the software development cycle will save time, money and energy spent on removing them after software release. We are developing continuous process for systematic and improvement of Software security throughout the various software lifecycle stages, that is suitable for industrial adoption, and focuses on preventing vulnerabilities in all phases of software development cycle. To aim of this paper we propose methodology for software security through Testing Stage of software development life cycle. We are enhancing the software through testing stage of SDLC. Testing is moderately expensive. Vulnerabilities Detection Tool is very good technique to cut down cost and time. Testing efficiency and effectiveness is the criteria for coverage-based testing techniques. Present tool well tested on Banking Domain Future work is to suit the tool for all domains.

IV. LIMITATIONS OF EXISTING METHODOLOGIES

There are four tool or framework available related to secure software engineering .security aware software development life cycle (SaSDLC) tool, vulnerability analysis tool or vulnerability analysis database, comprehensive lightweight application security process (CLASP) and microsofts SDL. For SaSDLC i.e. suraksha need to know use case diagram because in this, functional requirements of the system are analyzed and captured using UML tool and methodology. This system can be secured only against known threats.

In Vulnerability analysis database when a new vulnerability is discovered, a new iteration of process is initiated in order to prevent the vulnerability. When a known vulnerability recurs, this indicates that the existing vulnerability or activity models are incomplete. One of the most important limitations of this method is that they are inflexible and provide little or no support for evolution. Another common drawback is lack of specificity.

CLASP is a lightweight process for building secure software. CLASP is defined as a set of independent activities that have to be integrated in the development process. This does not determine whether the application is covered by methodology. it does not determine the bug bar SDL not having facility of data collection and reporting .periodically collect and evaluate is also not possible in SDL.

V. OBJECTIVE AND DATA COLLECTION

During the study of literature and previous research we found the following conclusion that leads to design a new concept of providing the security in the software development life cycle. The key factors that are found are listed below.

1. The software systems development is depends on the stack of different processes where all elements of stack (activity) are dependent on the previous activity.
2. Any gap, conflict, ambiguity or security gap can destroy all the system or make failure of the system.
3. Vulnerable or problems are variable for each and every project, thus preparation of individual policies are required.
4. To make more stability additional efforts and check list for gaps are required.
5. One main and most big challenge is ad hoc nature of projects which is variable according to execution environment, requirement gathering, and other phases of the SDLC.

Thus required to find a most optimum way to recover the problems and vulnerabilities in any project, for that purpose we suggest the following solution of the proposed work.

Success of software depends on the extent to which it meets stakeholder expectations. Requirement capture and analysis help in identifying stakeholders and their expectations, and capturing these expectations in a form that is amenable to analysis and implementation. Software projects fail to meet their expectations due to problems in articulation of requirements, poor quality of analysis and quite often, a lack of sufficient focus on the business perspective. A framework to remove vulnerabilities: analysis phase in SDLC will find out bugs or errors and missing requirements for analysis.

Thus we starts with analysis phase if the software development life cycle. Project requirement gathering involve the analysis of requirement, that leads to understand the problem identification and available resources in terms of (what actually required by the client), and similarly what gaps or vulnerabilities are occurred or possible in the project.

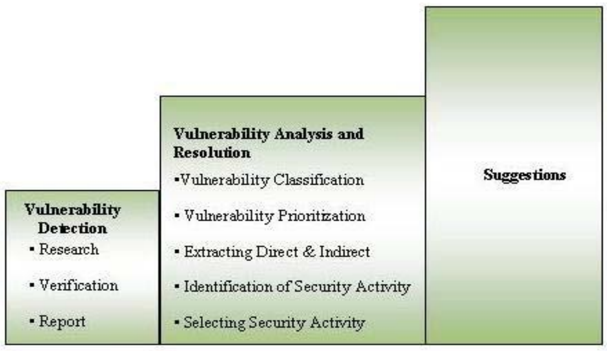
Vulnerability Detection
• Research
• Verification
• Report
Vulnerability Analysis and Resolution
•Vulnerability Classification
• Vulnerability Prioritization
• Extracting Direct & Indirect
• Identification of Security Activity
• Selecting Security Activity
Suggestions

Fig 4 Activities in Vulnerability Life Cycle

But not all the customers are having the sufficient knowledge or technical knowledge to explain the proper requirement of project, thus proper analysis can improve the project gaps.

In order to prevent vulnerabilities in the software under development, the VLC [1] is designed keeping the following objective in mind:

1. Detection of the vulnerabilities.
2. Classification of the detected vulnerabilities.
3. Identification of recurring vulnerabilities.
4. Determination of direct and indirect causes.
5. Listing the activities to resolve the causes of the vulnerability or vulnerability itself.
6. Selection of the optimal set of activities.
7. Suggestions.

To achieve the above objective, the vulnerability life cycle of VLC is divided into three major phases. Each phase of Vulnerability Life Cycle performs activities as given Figure. Due to consideration of security from initial phase of software development costing of software will less. Because of this new methodology less time is necessary. Due to this manual work is reduced because software fed all requirements to the next phase of SDLC, Due to this methodology analyst having less work because of fulfillment of all requirements.

VI. CONCLUSION AND FUTURE WORK

Software engineering is a large and complex domain of study, and the opinion about the SDLC and its phases are different according to different authors and researchers. in this paper we are study about the software development life cycle and their impact on different activities involved in the software development life cycle.

Additionally in this paper we discuss about the different aspects of providing security in software development stages, and provide the evidence where concluding facts are indicates about the policies to apply in the secure software development, after concluding the facts about security and Vulnerabilities we propose a new approach to identify the problems in initial stage, by which we take advantage of low maintenance, cost and effort for any kind of gap or Vulnerabilities.

In future we design, and implement the proposed technique for making and find Vulnerabilities in software development life cycle in analysis phase using the visual studio dot net framework which provide a secure, programmer friendly environment for development.

ACKNOWLEDGEMENT

Thanks to the professors of our institutution that have helped in Research.

REFERENCES

1. A Framework to Detect and Analyze Software Vulnerabilities-Development Phase Perspective, International Journal of Recent Trends in Engineering Vol 2, No. 2, November 2009
2. The Role of the Security Analyst in the Systems Development Life Cycle, SANS Institute InfoSec Reading Room, Brad Gray, MBA GIAC Security Essentials Certification (GSEC Practical Assignment) January 12, 2005.
3. NEXT GENERATION SOFTWARE SECURITY THROUGH TESTING STAGE OF SDLC, Vidyabhushan A. Upadhye1 and Shashank D. Joshi, IJCSC Vol. 2, No. 2, July-December 2011, pp. 311-313
4. REVIEW ON COMMON CRITERIA AS A SECURE SOFTWARE DEVELOPMENT MODEL, international Journal of Computer Science & Information Technology (IJCSIT) Vol 4, No 2, April 2012, DOI: 10.5121/ijcsit.2012.4207 83
5. Baking in Security During the Systems Development Life Cycle, CROSSTALK The Journal of Defense Software Engineering, March 2007
6. SOURCE CODE ANALYSIS TO REMOVE SECURITY VULNERABILITIES IN JAVA SOCKET PROGRAMS: A CASE STUDY, International Journal of Network Security & Its Applications (IJNSA), Vol.5, No.1, January 2013
7. Software Vulnerabilities, Banking Threats, Botnets and Malware Self-Protection Technologies, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 1, January 2011 ISSN (Online): 1694-0814 www.IJCSI.org
8. Design and Development of Software for Launcher Control System, Department of Computer Engineering and Information Technology College of Engineering, Pune - 411005. June 2012

AUTHORS PROFILE

Sunil Patel graduated from Department of Computer Science & Engineering at Central India Institute of Technology, Indore and M.Tech Scholar at CIIT, Indore and published his research work in 1 Research Paper in Conference at Indore and gave presentations in many of the institutions in Indore like RIT Indore, CIIT Indore, OIST, Indore etc.., and he conquered his major research work in Software Engineering from last Two years.

Deepak Kulhare Associate Professor of Computer Science & Engineering, CIIT Indore, and published many Research Paper in prestigious Conference, Jounals.He gave Guidance to many M.Tech Scholars.His major research work in Software Engineering.

Arif Khan Assistant Professor of Computer Science & Engineering, CIIT Indore, and published many Research Paper in prestigious Conference, Jounals.He gave Guidence to many M.Tech Scholars.