Rai University AHMEDABAD
RISING - A Journal of Researchers
Bi-Annuals Peer Review Journal
Vol.9 Issue 1 (Jan 2025)
Impact Factor : 5.999
RISING A journal of researchers
ISSN No : 2456-7752

RISK MANAGEMENT APPROACH IN THE SOFTWARE DEVELOPMENT LIFE CYCLE

Dr. Irfan Ahmed Khan

Assistant Professor, Rai School of Engineering, Rai University, Ahmedabad
irfanahmad.khan@raiuniversity.edu

ABSTRACT:

Risk and its management is an area based on the hypothesis of probability. It is well known that requirement and design phases of software development life cycle are the phase where security integration yields maximum benefits. In this paper we have tried to tie software security and software risk in a single string. It is a complete process that will help a developer to choose most appropriate risk management plan for giving software more security. As we know that a software development life cycle is used to make understand the researchers, scientists, project managers, programmers, working of particular software in an easier manner. Actually, software development life cycle gives a basic understanding about the start of a project. This includes a number of phases that provides sequencing of activities and these activities will perform during the implementation of required software. In this paper, a life cycle is proposed which will help developer to identify and mitigate risks at the early stage of development.

KEYWORDS: Software Security, Software risk, RMMM plan, Software development life cycle (SDLC), Software quality.

INTRODUCTION

In modern information era, information systems and networks frequently consist of software systems running on multiple Interconnected computers with diverse capabilities, such as servers, desktops, laptops, PDAs, and even cell phones In these systems, networking has become more critical than ever before [1]. Fast data sharing has been made possible by connectivity, but it also increases the risk of personal data hacking and attacks. The situation is further complicated by the growing complexity and extensibility of software systems, which increase the likelihood of security breaches and increase the susceptibility of information systems to attacks and failures. Since reactive network-based security techniques, such as firewalls and signature-based anti-spyware, have been proven to be ineffective in achieving secure software, software security—the concept of designing software so that it can operate correctly and continuously under malicious attacks [2]—has garnered a lot of attention lately. The process of planning when software risk is detected is known as software security. Worries about potential future events. It could occur or it could not. Although a lot of work has been done on risk monitoring and mitigation, an RMMM plan life cycle for protecting software throughout the design phase has not yet been identified.

This work has been done to highlight the impact of software risk on object oriented design when designing software [3, 4, 5]. In this work, it has to be identified first that what is software security. The third portion provides an overview of risk; the fourth section outlines a risk management life cycle; and the last section concludes with future work and its findings.

SOFTWARE SECURITY

Security is linked to the concept of safety, secrecy, and reliability. Number of security flaws and vulnerabilities exists owing to the shortcomings of security architecture and security mechanism [6]. Hackers and attackers target software flaws and take advantage of them rather than creating security vulnerabilities. Hacking should be made extremely difficult in order to preserve software security during the development phases [7]. Software security is concerned with protecting the application program; security architecture must be designed to meet the needs of product security goals and sensitive information contained therein; it encourages developers to create secure software that performs better under conditions created by malicious attacks. The goal of making software secure is to protect the software from the various types of attacks, errors, bugs, threats, viruses, and vulnerabilities [8].

WHAT IS RISK?

Risk is future uncertain events with a likelihood of occurrence and a potential for loss. Risk is the expectation of the loss or damage. Quantifying the degree of uncertainty and loss associated with each risk is crucial when analyzing them [9]. The element that needs to be determined before examining software security is risk. Risks can be essentially split in two categories which are proactive risks and reactive risks. Proactive risks are the pre assumptions that dangers are to be happening in future. Figure 1 above illustrates how risk works in a software context. The figure illustrates how scheduling, hardware, system, technology, people, and cost can all be subject to risk. Before software is developed, these kinds of hazards are taken into account and planned for. When hazards are recognized early in the design process, secure software is created. Reactive risks arise when an issue arises following software deployment. Today's internet-dependent world requires secure software, which is only safe when its hazards are recognized early on and controlled. The essential component of safe software is risk identification, mitigation, and monitoring.

The process of recognizing, addressing, and removing hazards before they have a chance to harm the project is known as risk management. It recognizes software risks, makes measures to prevent them, and lessens their impact should they materialize. Although it is impossible to completely eliminate all hazards, we may try to reduce them by using risk management.

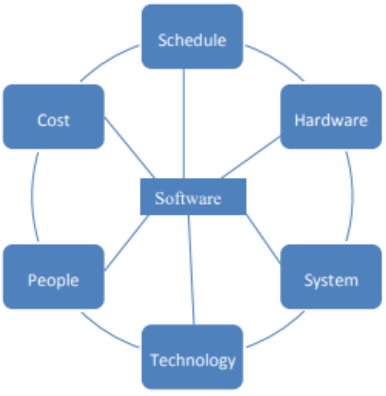

Fig: Risk within software context

RISK MANAGEMENT

Risk management is the identification, assessment, and prioritization of risks followed by the coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities [10-11]. In the Software Development Life Cycle (SDLC), "Risk is the prospect of suffering failure." Failure in a software development project refers to a negative impact to the project, which could be in the form of diminished quality of the end project increased costs, postponed completion, or complete project failure.

In the proposed framework of risk management, a life cycle is presented to identify and mitigate risks during the software development phase [12-13]. This life cycle of risk management is described with fig 2 and in detail; it is evident from the figure that risk management activities involve the six phases: Requirement phase, Analysis phase, Design phase, Development phase, Test phase, and Maintenance phase. Risk management activities consists of two major activities: Risk assessment and risk reporting, where risk assessment activity includes Risk identification, Risk analysis, and Risk Prioritization.

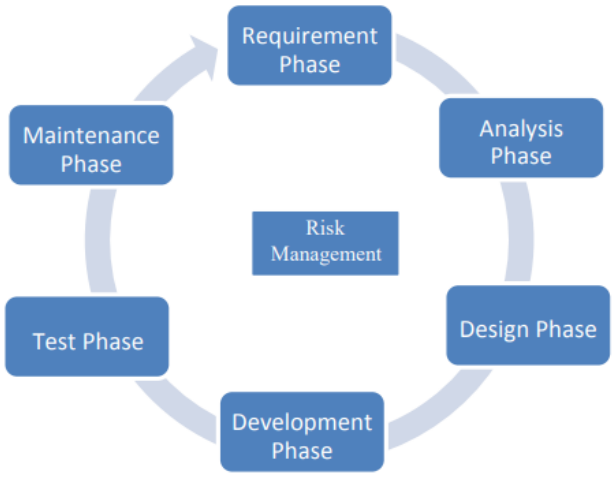

Fig2: Risk management activity in SDLC

Risk management activity in SDLC phases in details are as follows:-

A. Requirement phase •

User requirements are collected during the SDLC's requirement phase. Since hazards can arise throughout any stage of the SDLC, including the requirements phase, they are recognized and evaluated here. This phase involves two processes:

• Identification of assets: - Assessment of the probability of specific disruptions and the controls to reduce the risk to the organization. Alongside the vulnerability assessment, it is carried out [14-15].

• Threat identification: During the requirement phase, this procedure is utilized to find threats. To identify hazards and direct ensuing design, coding, and testing decisions, threat analysis approach is employed. Since many systems have specific requirements that produce unique vulnerabilities, identifying security threats is a planned process that calls for some imagination [16].

B. Analysis phase

A developer must comprehend the information domain's necessary functions, actions, and plans for risk mitigation in order to comprehend the nature of the risk report collected in the first step. Evaluating each risk item's loss probability and magnitude is the aim of the analysis phase.

C. Design phase

To find security issues and privacy risks, thoroughly examine security and privacy requirements and expectations during the design stage of development. Identifying and addressing these risks and concerns during the design phase is efficient. This process includes following steps.

- Asset identification: - Iterates through the capabilities and assets. Determine all potential security dangers for every security service on every capacity.
- Vulnerability identification: - A part of the enterprise risk management process is a security assessment or security vulnerability analysis. This step identifies vulnerabilities that arise from interactions with other systems or that are present in the software environment.
- Risk assessment: - Following identification, risks need to be evaluated for likelihood of occurrence and potential impact severity. It may be easy to measure these quantities or

impossible to detect their occurrence. Making the most informed choices during the assessment process is therefore essential to appropriately prioritizing the risk management plan's implementation.

- Risk mitigation: - A plan that would lower or eliminate the risks that are most important is known as risk mitigation. The mitigation plan designates a principal handler for each action and outlines the steps that can be performed to reduce the red-rated risk.
- Test plan and development: - A test and development strategy should be created in order to prepare for the next phase, and any associated risks should be recognized.

D. Development phase –

Building the code and documentation for the solution components is the main objective during the development phase. Throughout the phase, the team keeps track of every risk and deals with any new ones that come up. There are three steps in this process.

- Code reviews - A code review can be a useful tool for a team to determine whether code satisfies local standards. It may even uncover issues before compilation, which could pose future dangers [17].
- Pair programming: - Staff-loss risk is decreased through pair programming [18]. The pair programming shoulder-to-shoulder technique results in the highest rates of problem eradication by acting as an ongoing design and code review.
- Unit testing and static testing: - Developers can confirm that the countermeasures being created eliminate any security risks previously discovered through threat modelling and source code analysis, as well as validate the security functioning of components, by employing unit tests and dynamic analysis.

E. Test phase

- Dynamic code testing: Dynamic code testing is the process of analyzing computer programs by running them in a real or virtual environment. Enough test inputs must be used when running the target program in order to generate intriguing behavior.
- Web application testing: - Thorough testing of a web-based system prior to launch can assist in resolving problems before they are made public. Issues include the web application's availability, authorization, and security, among others.

- Vulnerability scanning: - Vulnerability, which comprises testing space scanning and non-defect scanning, is a crucial technique for identifying software security threats. Network port, string, producer data, network data, and other element scanning are all included in testing space scanning. Non-defect scanning often uses the defect library to identify non-flaws.
- Test threat actions: - Threat has a detrimental impact on the test. Therefore, testing threats generates risks, and this September is taken into consideration to identify those risks.

F. Deployment phase

The product is only partially finished during the deployment phase. At this stage, a suitable test strategy is created after all risks have been recognized throughout the life cycle.

- Periodic testing: - Periodic testing refers to the requirement for ongoing software development to undergo third-party testing.
- Risk management plan: - Creating a risk categorization table, ranking the risks, creating and organizing the risk table, and making sure that risk management is a continuous process throughout the project are all simple tasks in creating a risk management plan.

FUTURE WORK

Since half of software is known to be planned and developed in paper form, taking risk into account when working on paper will reduce risk on its own and minimize it. A proactive strategy that closely monitors security at every stage avoids costly during the design stage of the SDLC, security needs and features are crucial to the integration of security. The next step for this suggested framework is to plan an RMMM design that will be typical for certain risk types.

CONCLUSION

Risk management is a methodical way to see the risks to the success of a project. We can ensure that the most severe risk is controlled first by taking into account the possible effects of each risk item. We cannot guarantee that our risk management activities are carried out correctly without a systematic methodology. Therefore, the document, which offers a step-by-step implementation of a risk management plan, justifies a suitable risk management plan life cycle.

REFERENCES:

1. M. Howard and D. LeBlanc, "Writing Secure Code", Microsoft Press, 2001.
2. Gary McGraw, "Software Security", IEEE Security & Privacy, vol. 2(2), 2004, pp. 8083
3. J. Viega and G. McGraw, "Building Secure Software", addition Wesley, 2001.
4. S. Chandra and R. A. Khan, "Object Oriented Software Security Estimation Life Cycle: Design phase perspective", Journal of Software Engineering, USA, pp: 39-46.
5. Roger S Pressmen "Software Engineering a Practitioner's approach", Book.
6. http://en.wikipedia.org/wiki/pairprogramming.
7. Introduction to software testing available at http://www.onestoptesting.com/introduction/
8. http://technet.microsoft.com/en-us/librry/bb497041.aspx
9. G. Booch, Object-Oriented Analysis and Design: with applications. Benjamin/Cummings, 1994.
10. E. S. Ochotta, et al, "A Novel Predictable Segmented FPGA Routing Architecture," in FPGA '98, Proceedings of 1998 ACM/SIGDA intl. symp. On FPGAs, pp. 3-11.
11. C.W. Krueger, "Software Reuse." ACM Computing Survey, vol. 24, no. 2, pp. 131-182, 1992.
12. E. Mettala and M.H. Graham, "The Domain-Specific Software Architecture Program," CMU/SEI-
13. E. S. Ochotta, et al, "A Novel Predictable Segmented FPGA Routing Architecture," in FPGA '98, Proceedings of 1998 ACM/SIGDA intl. symp. On FPGAs, pp. 3-11.
14. J. Lakos, "Large-Scale C++ Software Design," Addison-Wesley Professional Computing Series
15. http://www.xilinx.com/company
16. M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA, 1979.
17. J. Frankle, "Iterative and Adaptive Slack Allocation for Performance-driven Layout and FPGA Routing," Proceedings of the 29th ACM/IEEE conference on Design automation conference, 1992, Page 536.