Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 2 (2017) pp. 203-210

http://www.ripublication.com

Aspects of Enhancing Security in Software Development Life Cycle

Anuradha Sharma[1] and Praveen Kumar Misra[2*]

[1] Dept. of Computer Science, Amity University Lucknow (U.P.), India.

[2] Dept. of Mathematics and Statistics, DSMNR University, Lucknow (U.P.), India.
**Corresponding Author*

Abstract

Softwares have become an integral part of everyday life. Every day, millions of people perform transaction through various applications run by these softwares as internet, ATM, mobile phone, email etc. Softwares are used by people bearing in mind that it is reliable and can be trusted upon and the operations they perform is secure. Now, if these softwares have ensembled security holes, then how can they be safe for use? Security brings value to software in terms of people's trust. The value provided by secure software is of vital importance because many critical functions are entirely dependent on the software. Because of the limitation of budget and release time of the software into the market, many developers consider security as an afterthought thus resulting in poor quality software. In the early days, software security was only considered as part of software testing but, later on, it has been experienced that security is not an afterthought in case of software development[16][17]. Various aspects of securing the software by enforcing security in various phases of software development life cycle have been looked upon by referencing novel work by various authors on security in SDLC.

Keywords: Software security, SDLC, Neural Networks.

1. INTRODUCTION

Software security is the idea of software engineering to protect the software from the unauthorized access and also protect from the malicious attack. Software security makes the system software function continuously and correctly under malicious attack [1]. There is various approach currently used for integrating security in software development life cycle model. Implementing security from earlier stage of the software development makes the system fault free as possible and less vulnerable. As the day by day technology increases there is wide use of software which increases the security threat of software. Various security attacks due to security flaws in software harm the organization and also affect the financial status as well as integrity of the organization. Security is not a unique feature of software; it is an important part of the software which is to be done carefully during the development of the software. Application security aspect must be integrated in the software development process. After development, process testing is not sufficient because it's too late to fix the bugs and mistake. Security is implemented at every major phase of software development life cycle. There are various approach used to integrate the security at various level of SDLC. Some of the author's focused on the security at the initial phase(requirement & Design phase) and some at the other half (coding & testing) of the development phases. [11]

2. RELATED WORK ON THE ASSESMENT OF SECURITY IN SDLC MODEL

In order to design software more secure many approaches have been adopted at the various level of software development life cycle model. Some of these approaches are given below.

2.1 Software Security Rules

Most familiar way to spread software security knowledge is to train software development staff on critical software security issues. Beyond responsiveness, advanced software security training should offer coverage of security engineering. Many types of research is done by researchers in this direction but there are so many research issues that need to be addressed. There are twenty one rules that are proposed in this paper to make software more secure. If these security rules are followed properly, it will help the security mechanism. These rules protect the software from unauthorized access; protect software from being infected, provide access control of the software, help to make software more accurate and consistent, also helps to improve the interoperability [1]

2.2 Practicing Best Security Practices

The importance of software security has been felt by practitioners and as a result, adaptation of best practices to address such problems has also increased. "Precaution

is better than cure", it is a common saying and is very much applicable in case of ensuring software security. One has to be very careful in inculcating the aspect of security deeply during the development process so that the experience of these practitioners over the past years can be availed positively and software security can be assured by assuring secure development process. How software practitioners can apply them to the various software artifacts produced during software development. In the rest of this section, we will touch on best practices. As this department unfolds, we will cover each of these areas in much greater detail.

Security should be explicitly at the requirements level. Security requirements must cover both functional security (say, the use of applied cryptography) and emergent characteristics. One great way to cover the emergent security space is to build abuse cases. At the design and architecture level, a system must be coherent and present a unified security architecture that takes into account security principles (such as the principle of least privilege). Designers, architects, and analysts must clearly document assumptions and identify possible attacks. At both the specifications based architecture stage and at the class-hierarchy design stage, risk analysis is a necessity. Security analysts should uncover and rank risks so that mitigation can begin. External review by external people not belonging to the design team is often necessary.

At the code level, one should ensure the flawless implementation. The flaws in implementation can be discovered by static analysis tools. These are the tools that scan source code for common vulnerabilities.

Security testing must encompass two strategies: testing security functionality with standard functional testing techniques, and risk-based security testing based on attack patterns and threat models. Any good security test plan having traceability of back to requirements will use both of these strategies. So standard-issue quality assurance is unlikely to uncover all the pressing security issues. Penetration testing can be very useful, especially if an architectural risk analysis is specifically driving the tests. The penetration testing is advantageous because it results in a good understanding of fielded software in its real environment. Softwares that are victim of canned black-box testing are very poor in practice. This is reason that the black box testing is a very simplistic application security testing tool available in the market. This does not ensure software security at a greater depth. People involved in operations should carefully monitor fielded systems during use for security breaks.

Knowledge gained by understanding attacks and exploits should be cycled back into the development organization, and security practitioners should explicitly note that risks crop up during all stages of the software life cycle. So, a constant risk analysis thread, with recurring risk tracking and monitoring activities, is highly recommended[13][14].

Threat modeling is a very important activity that is carried out at the development phase to protect the software application from threats and vulnerability to ensure a more adequate sense of its security [2]. A software system can be influenced by vulnerabilities, threats, attacks and countermeasures. The technique of threat modeling can be used to identify all/any of these [3]. Threat modeling helps to

understand how a malicious attacker chooses targets, locates entry points and conducts attacks [4]. Through threat modeling, one can address the threats that can cause maximum damage to any software application.

The key to threat modeling is to measure where most effort is needed to make system software more secure. [15]

Architectural risk analysis is a technique that makes it possible to identify vulnerabilities and threats at the design phase of the software development life cycle. The nature of the software system decides the nature of the threat and vulnerability as malicious or non-malicious. It examines the preconditions that must be present for the vulnerabilities to be exploited by various threats. The advantages of architectural risk analysis is that it enables developers to analysis software system from its component level to its environmental level in order to evaluate the threats, vulnerabilities and shock at each level.[5]

Attack trees approach is used to characterize the security of the system. In this approach attackers tree is generated and the root of the tree is represent the goal of attackers. The node of the tree represent the different type of action is taken by the attackers to accomplish his goal. The attack tree is used to analyze the risk and also in design, implement and measure the attack. So this approach is basically used in design phase of the SDLC. [6]

Attack nets are the similar approach to attack tree which include places equivalent to node in attack tree to point out the state of attack. In this when event move from one place to another place are captured in

the transition and the transition point out the path of an attacker takes. [7]

The vulnerability tree is an approach where tree is constructed and on the basis of this tree we find out the how one vulnerability relates to another and the steps an attacker may take to reach the top of the tree. Vulnerability trees help to analyze the different possible attack scenario that attackers can take to exploit the vulnerability. [8]

Gegick and Williams also proposed a regular expression-based attack pattern [9] which helps in representing the chronological events that occur during an attack. In this approach software component is used to identifying the vulnerability in software design. It consists of attack library of abstraction which is used by software engineers conducting Security Analysis for Existing Threats to match their system design. If a match found then it indicates that vulnerability exits in the system being analysed and helps to overcome these flaw before coding is started. This approach can be easily adapted by developers who are beginners on security.

The neural network approach is an approach where security flaws is analyzed in software design based on the abstract and matching technique through which software flaws can be easily identified when attack pattern is matched to the design. The Williams and Gegick [9] proposed regularly expressed attack pattern that is used to identify the actor and software components. Online vulnerability database is used to identify the attack scenario corresponding to attack pattern. Three layer feed forward

back propagation is used for the architecture of the neural network. A training data is passed as an input and the input is match to online vulnerability data base. [10][12]

2.3 Comparative Study

As per literature survey we have studied many papers on the security of the software development life cycle model which have been compared and tabulated below which will be useful for the future research. There are various paper techniques available in this area. So we are doing comparative study of security in SDLC model based on the advantages and limitations.

Table 1: Techniques for Security in SDLC

Security in SDLC (Techniques)	Advantages	Limitations
Software security rules	• 21 rules to make software more secure. • Minimize the risk level.	• These rules further divided in to sub rule and make software more secure
Threat modeling	• It helps to identify the threat, attack and vulnerability in the software	• It is limited to only design phase of the SDLC.
Attack tree	• It is helpful in risk analysis and also in threat detection by drawing tree.	• It is also limited to only design phase • it is very tough to implement for large project
Attack network	• It is used for threat assessment when event move from one place to another place	• Used only at design level and attack net. Must be drawn before implementing any security assessment
Vulnerability tree	• It is used to identify the vulnerability through vulnerability tree • It helps to analyze different possible attack scenario	• It is limited to vulnerability hierarchy tree which is constructed from relation of one vulnerability to another • It is very tough to find out the relation between vulnerability
Architectural risk Analysis	• used to identify vulnerabilities and threats at the design phase of SDLC	• it can be used only in design phase

	• advantages of architectural risk analysis is that it enables developers to analysis software system from its component level to its environmental level in order to evaluate the vulnerabilities, threats and impacts at each level	
Neural network approach	• Is used to identify the threat, attack, vulnerability in software design phase	• Online vulnerability data base and training data is always required to perform this approach.
Gegick& Williams regular expression-based attack	• proposed a regular expression-based attack patterns which helps in indicating the sequential events that occur during an attack • Software components involved in an attack and are used for identifying vulnerabilities in software designs. • It can be easily adapted by developers who are beginner on security.	• It is limited to attack library which is always needed to identify the vulnerability. • To make attack library, a lot of experience required which is very expensive and time taking.

CONCLUSION

The paper represents an overview of various security models and their performance. The entire models discussed above have many advantages and limitation. We have studied various model and their methodologies to make software more secure and to protect software from unauthorized access. Most of authors have emphasized on the security at development phase (design). The comparative study is done on the basis of advantage and limitations of the proposed model. This paper can be useful for the further research.

REFERENCES

[1] C. Banerjee, S. K. Pandey, "Software Security Rules: SDLC Perspective", (IJCSIS) International Journal of Computer Science and Information Security, Vol. 6, No.1, 2009

[2] Agarwal, A. 2006), "How to integrate security into your SDLC", Available at: http://searchsoftwarequality.techtarget.com/tip/0,289483,sid92_gci1174897,00.html,

[3] Meier, J. D., Mackman, A. And Wastell, B.(2005), "Threat modelling web applications", Available at: http://msdn.microsoft.com/enus/library/ms 978516.aspx

[4] Redwine, S. T. Jr and Davis, N.; et al, (2004), "Process to produce secure software: Towards more secure software", National Cyber Security Summit, Vol. 1

[5] McGraw, G. (2006), "Software security: building security in", Addison-Wesley, Boston, MA

[6] Redwine, S. T. Jr and Davis, N.; et al, (2004), "Process to produce secure software: Towards more secure software", National Cyber Security Summit, Vol. 1

[7] Gegick, M. and Williams, L. (2006), "On the design of more secure software-intensive systems by use of attack patterns", Information and Software Technology, Vol. 49, pp 381-397.

[8] Ralston, P.A.S; Graham, J.H and Hieb, J. L. (2007), "Cyber security risk assessment for SCADA and DCS networks", ISA Transaction, Vol.46(4), pp583-594

[9] Gegick, M. and Williams, L. (2006), "On the design of more secure software-intensive systems by use of attack patterns", Information and Software Technology.

[10] Security Assessment of Software Design using Neural Network A. Adebiyi, Johnnes Arreymbi and Chris Imafidon School of Architecture, Computing and Engineering University of East London, London, UK.

[11] Julia H. Allen, Sean Barnum, Robert J. Ellison, Gary McGraw,Nancy R. Mead: Software Security Engineering: A Guide for Project Managers, Addison Wesley Professional, 2008, pp 6-8.

[12] Srinivasa, K.D. and Sattipalli, A. R, (2009), "Hand written character recognition using back propagation network", Journal of Theoretical and Applied Information Technology, Vol. 5(3), pp 257-269

[13] J. Wing, "A Call to Action: Look Beyond the Horizon," *IEEE Security & Privacy*, vol. 1, no. 6, 2003,.

[14] G. McGraw, "Building Secure Software: Better than Protecting Bad Software(Point/Counterpoint with Greg Hoglund)," *IEEE Software,* vol. 19, no. 6, 2002, .

[15] Anurag Agarwal, ―Threat modeling enhanced with misusecases, searchsoftwarequalitytechtarget.comhttp://searchsoftwarequality.techtarget.com/t.html,Aug.2,2008.

[16] Anuradha Sharma, Dr. Praveen Kumar Misra, "Secure Software Requirement Specification", presented and published at International Conference on Recent Trends in Engineering, Technology and Management, Bundelkhand Institute of Engineering and Technology,2011, ISSN 978-93-80697-69-7.

[17] available at https://www.cigital.com/blog/what-is-the-secure-software-development-lifecycle/, 2016.