Secure Software Development Lifecycle

Rajeshwari K Gundla

Senior Faculty
School of Information Technology,
Ajeenkya D Y Patil University
Pune, India

***Abstract*: Every software goes through lifecycle which consist of different stages. For building secure software security aspect has to be kept in mind at each stage. The Objective of the software is to me customer requirements but CIA (confidentiality, Integrity and availability) aspect of the security for software must not be ignored. The Programmer or Developer writes program code to accomplish a particular task. While writing program, developer / programmer don't consider vulnerabilities in the program which can be helpful to attacker to get into the system and perform malicious activities which can cause DAD (Disclosure, Alteration and Denial) of Data. Software should be in such a way that it should not contain any loopholes. Attackers write malware programs whose task is to find vulnerabilities in genuine program and get into the system and do malicious tasks ranging like spreading into the network, dropping a malicious file in the system, and many more. In this paper, we are proposing how to build secure software lifecycle by including security aspect at every stage.**

***Index Terms*: CIA & DAD triad, Secure SDLC, Code Review**

I. Introduction

Any software passes through different stages during its lifecycle [1]. In the preliminary stage customer requirements of software are gathered after that key decisions on software design and architecture are made. Based on design and architecture, implementation i.e. coding is carried out. Whether customer requirements are fulfilled or not is checked in testing stage [2]. After testing deployment of software is done at customer place. If there is any malfunctioning or updates software maintenance is carried out. Fig.1 is the flowchart of software development lifecycle [3].

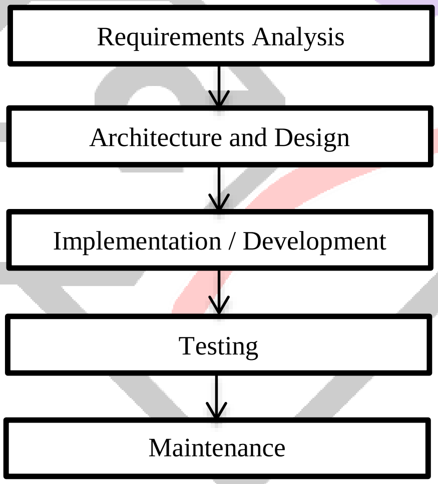

Fig.1: Existing Software Development Lifecycle

Along with fulfillment of customer requirements, major aspect of security which is CIA (confidentiality, Integrity and availability) must be considered at each stage of lifecycle. Confidentiality means only authorized person can access software application. Integrity means only authorized person can modify and update software application or data. Availability means data should be available to authorized user when needed. Cybercrimes are increasing day by day. Attackers are finding loopholes in the existing software applications to find a path to enter into the system to do their intended tasks such as DAD (Disclosure, Alteration and Denial). Disclosure meaning gaining access to software application or sensitive data without permission. Alteration means modifying or updating data without permission. Denial Means interrupting the availability of information or software application [4]. As shown in Fig.1 Existing Software development lifecycle does not include security aspect at each stage, hence there is need for modifying lifecycle of software development to include security aspect at each stage.

II. Methodology

Considering security while developing software application is a need of an hour [5]. Secure software development lifecycle puts emphasis on security at each stage of lifecycle. Fig.2 shows secure software development lifecycle (Secure SDLC).

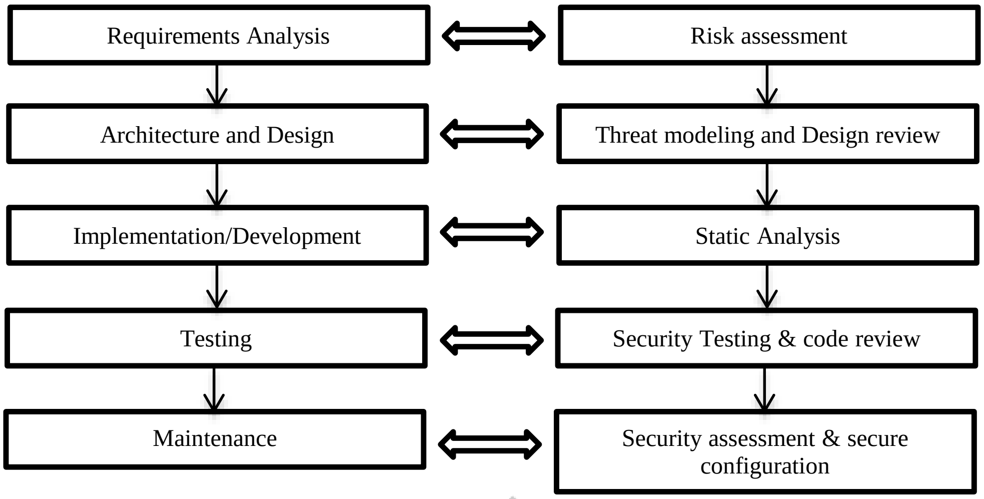
Requirements Analysis
Risk assessment
Architecture and Design
Threat modeling and Design review
Implementation/Development
Static Analysis
Testing
Security Testing & code review
Maintenance
Security assessment & secure configuration

Fig.2 Secure software development lifecycle (SDLC)

1. Risk Assessment

To meet CIA conditions there must be understanding of risk faced by organization. Every organization can have unique way of operation therefore risk faced by organization also vary. There are three major factors in risk assessment vulnerabilities, threats and risk. Vulnerability means weaknesses in software application that might allow breach of CIA. Threats are the external entities who try breach security. Risk is the security breach incident caused by threats by exploiting system vulnerabilities. Fig.3 shows the relationship between threats, vulnerabilities and risks.

Threat
+
Vulnerability
=
Risk

Fig.3 Relationship between threats, vulnerabilities and risks

2. Threat Modeling and Design Review

Threat modeling is a computer security optimization process that allows for a structured approach while properly identifying and addressing system threats. The process involves systematically identifying security threats and rating them according to severity and level of occurrence probability.

Benefits of Threat Modelling (architectural risk analysis):-

- Identification of Biggest threats
- Having plan on identified and documented threats
- Elimination of security issues in design phase
- Prioritization of development and testing based on identified threats
- Optimization of cost in effective manner

For building secure applications you first need to understand the common design flaws

Common design flaws:-

a. Insufficient authentication and authorization
b. Broken session management
c. Insecure external components

The following four point’s framework gives the needed structure for threat modelling:

1. What are the things we want to protect and what are the entry, exit points for threats
2. What can compromise or damage your asset.
3. How to mitigate threats and reduce risk.
4. Validation of previous steps

3. Static Analysis

Static analysis means analyzing the source code of any software application without executing it. Static analysis is also known as White Box testing in this code review is done to see if the proper coding convention and standards are followed or not. In Static Code Analysis various tools can be used to highlight possible vulnerabilities within 'static' (non-running) source code. These techniques are often derived from compiler technologies. Following are the few techniques

3.1 Data Flow Analysis:

Data flow analysis collects run-time information about data in software while it is in a static state.

3.2 Taint Analysis:

Taint Analysis tries to identify variables that have been 'tainted' with user controllable input and traces them to possible vulnerable functions also known as a 'sink'. If the tainted variable gets passed to a sink without first being sanitized it is flagged as a vulnerability.

3.3 Lexical Analysis:

Lexical Analysis converts source code syntax into 'tokens' of information. These tokens can be checked for vulnerability

4. Security Testing

Security testing is done to test all possible scenarios uncover vulnerabilities of the system. It is done to protect system and data from threats. Security Testing is used to make sure that software application is free from any loopholes that cause big loss. Security testing checks all possible loopholes and system vulnerabilities which might cause data loss and risk to the organization.

Security testing involves vulnerability scanning, security scanning, penetration testing, risk assessment, security auditing, ethical hacking and posture assessment. Here security tester should play a role of attacker to find system vulnerabilities. Thus security testing is very important.

5. Security assessment

After deployment of software it is necessary to monitor software application to assess security posture. Assessment gives information on flaws of the system which can be improved. Security assessment also provides information about which aspects of the security system can be updated.

III. Conclusion

Considering security while developing software is very important aspect [6]. Normal software development lifecycle does not include security as a mandatory step. In this paper we have presented how security can be implemented at every stage of software development. Also we suggest developers to consider vulnerabilities in a software application while developing it so that it becomes difficult for attackers to find loopholes in system. Software application should be zero vulnerable.

IV. FUTURE SCOPE

Along with writing secure software application we are working on changing the compiler architecture so that compiler should give warning if it finds any suspicious or malicious code.

References

[1] Harshad S. Modi, Nikhil Kumar Singh, Harsha Pradeepbhai Chauhan, "Comprehensive Analysis of Software Development Life Cycle Models", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395 -0056, p-ISSN: 2395-0072, Volume: 04 Issue: 06 | June -2017

[2] Mohit Kumar Sharma, "A study of SDLC to develop well engineered software", International Journal of Advanced Research in Computer Science, ISSN No. 0976-5697, Volume 8, No. 3, March – April 2017

[3] Sahil Barjtya, Ankur Sharma, Usha Rani, "A detailed study of Software Development Life Cycle (SDLC) Models", International Journal Of Engineering And Computer Science, ISSN: 2319-7242, Volume 6 Issue 7 July 2017, Page No. 22097-22100

[4] David Seidl, Mike Chapple, James Michael Stewart, "Comptia Security+", Microsoft

[5] Hala Assal, Sonia Chiasson," Security in the Software Development Lifecycle", Fourteenth Symposium on Usable Privacy and Security. August 12–14, 2018, ISBN 978-1-931971-45-4

[6] https://www.techopedia.com/definition/22193/software-development-life-cycle-sdlc