Security During Secure Software Development Life Cycle (SSDLC)

Dr. Rajiv Chopra  
Ph. D. (Computer Science), Assistant Professor  
(CSE/IT)  
GTBIT, GGSIPU DELHI.

Dr. Sushila Madan  
Associate Professor  
LSR College, Delhi University  
Delhi

Abstract:  
Website security is an umbrella activity. This implies that it should be carried out throughout the Software Development Life Cycle (SDLC). It is only after that it will become secure software development life cycle (SSDLC). Both testing and security work in parallel. Hence, if website testing is done throughout SDLC then security is also improved at every phase of SSDLC.


1. Introduction
Companies understand the need of protecting data by securing web applications. But securing any web application does not mean slowing down of SDLC. The need is to amalgate both security and SDLC process at the very beginning of the software project. This mitigates (short-circuits) expensive and time consuming issues at the later stages of SDLC. This will help engineers to deliver end product at a faster rate and to reduce web application vulnerabilities. The delays in the project can also be avoided and cost be minimized. There has to be a trade-off between business needs, usability and security.

2. Literature Review
T. K. Wijayasiriwardhane, P. G. Wijayarathna and D. D. Karunarathna in [1] propose and develop a tool to automate the test case generation process. The tool is named as Path Tester. It takes as input a program in form of a flowchart and converts it into a corresponding flowgraph. It then determines the paths to test. The test case generator module is then invoked for each path stored in the basis set data structure. However, this tool cannot handle complex situations with control dependencies. It cannot handle multi-way selections but assumes only single point entry and exit.

Mohammad Bokhari and Shams Siddiqui [2] explain that with proper requirements analysis, a project can be delivered on time and within cost budgets. They also stress that as software vulnerabilities increase, additional security aspects are needed to make software more reliable and robust. But they have not proposed any new model or tool to test for the software vulnerabilities.

Vineet Tyagi, Devesh Kumar and Vijay Maheshwari [3] propose three dimensional security aspects—security by construction, security by verification and security by protection. It does not provide any solution model to solve such an ever growing cyber security issues.

Diana Kelley of Security Curve [4] suggests that web security is achievable in a time-sensitive manner when stakeholders work together collaboratively. A mature model is the one in which the people, the process and the technology are together integrated throughout SDLC.

3. Proposed Work
Traditional SDLC involves five phases of software development as follows:-
Phase-1: Requirements Analysis.  
Phase-2: Software Design.  
Phase-3: Software Coding.  
Phase-4: Software Testing.  
Phase-5: Software Maintenance.
But modern SSDLC must involve the following phases:

Phase-1: Requirements Analysis + Security.
Phase-3: Software Coding + Security.

Building security into SDLC to get SSDLC.

Furthermore, using static and dynamic assessment tools together enables managers and developers to prioritize which all applications, module and vulnerabilities are more accurate. So, they must be addressed first. It has been found that training and educating developers in every phase of SDLC is equally important. Train them about the IDEs, automated tools—both manual as well as online tools available in the market today. Some are free and some are paid tools. The objective is to weave security throughout.

**Phase-I: Requirement Analysis Security:**

a) Check for SRS (Software Requirements Specification) document. It should be in IEEE standard format. Review it. It should be complete, concise, unambiguous, consistent, verifiable, traceable, maintainable and so on.

b) Draw functional and non-functional diagrams like Use-Cases, DFDs, ER diagrams and UML diagrams.

c) Do proper documentations. Draw flowcharts for the problem in hand.

Advantage of doing this is decreased time-to-deployment, reduction in acceptance bottlenecks. Failing to test against requirements rarely results in the software product you intended. Introducing compliance requirements, agile processes, market pressures and risk management makes testing even more daunting. It is nearly impossible to verify requirements without connecting test cases, results and objectives. Traceability tools make the process more manageable while also ensuring all requirements are verified by proper tests.

**Phase-II: Software Design Security:**

a) Do risk management---risk identification, risk projection, risk assessment and risk management. Develop RMMP—RISK MANAGEMENT AND MONITORING PLANS.

b) Check for security during inter-operability between components, tiers or systems.

c) List down vulnerabilities that are observable till this point.

d) Draw dependency graphs between interactions.

e) Design security plans.

f) Perform SDD—Software Design Document-reviews.

Advantage of doing this are that risks are identified at an early stage, reusable use cases save time during testing phase and other design problems are also avoided.

**Phase-III: Software Coding Security:**

One very simple equation holds here:-

{Risk-assessed Design} + {Security Control} + {Automated Tools} = Better Code.

Perform:-

a) Code reviews.

b) Risk analysis.

c) Security team must check for components—meeting security needs.

d) Use Tools—code generators, SQL and other Fourth and Fifth generation techniques for coding.

Advantages of doing this are lesser vulnerable code is achieved, reusable code and lesser error code is obtained.

**Phase-IV: Testing Security:**

The following activities must be done:-

a) Dynamic security testing—test executables.

b) Do run-time analysis now.

c) Composite security analysis.

d) Both testing and security have symbiotic associations.
e) QA activities must be done.
f) Give priorities to the vulnerabilities found based on security issues.
Advantages are better communication and faster fixes.

**Phase-V: Maintenance Security:**
The following activities must be carried out:

a) Regression tests your web applications.
b) Monitor data leakages when data is transferred.
c) In-house/ out-house monitoring can be done.
d) This will find out hidden vulnerabilities also that were not found during earlier SDLC phases.
Advantages are continuous improvement (with feedback loops), hidden vulnerabilities are located (better KB) now.

**Security Tools:** IBM Rational AppScan, to develop iterative, repeatable solutions that support security and risk management, can be used throughout SDLC.

**Conclusions:** Web application security is achievable in a time-sensitive manner where entire SDLC team works in a collaborative manner. The need is to find the impact of feature changes on software and where to focus on testing, to provide traceability from requirements to testing and finally, to coding phase, to confirm requirements validation and verification and to use project templates is another objective. The following table shows different SDLC phases and the must activities to be carried out:

<table>
<thead>
<tr>
<th>Phases (s)</th>
<th>Activities to perform</th>
</tr>
</thead>
</table>
| 1. Requirement Analysis | • Interviews.  
|                        | • Use cases.  
|                        | • Definition.  
|                        | • Misuse cases.  
|                        | • Security requirements.  
|                        | • SRS review.  
|                        | • Non-functional testing.  
|                        | • Static testing.  |
| 2. Software Design     | • Components  
|                        | • Relationships.  
|                        | • Flowcharts.  
|                        | • Risk assessments.  
|                        | • SDD review  
|                        | • Reusable Security Plans.  |
| 3. Software Coding     | • TDD (test before development) may be used[5].  
|                        | • Risk analysis.  
|                        | • Code reviews.  |
| 4. Software Testing    | • Functional testing.  
|                        | • Dynamic testing of security.  
|                        | • Components security.  
|                        | • Risk analysis.  |
| 5. Software Maintenance| • Runtime Monitoring.  
|                        | • Security Monitoring.  
|                        | • Regression Testing.  |

**Table-1: SDLC phases with basic Testing and Security Activities**
References


