Security Considerations for the Spiral Development Model

Loye Lynn Ray
University of Maryland University College
3501 University Blvd East
Adelphi, MD 20783
Loye.ray@faculty.umuc.edu
717-718-5727

Abstract

Security measures have been well established within the waterfall development life cycle model. However, more software projects are using the spiral development life cycle model. Software developers and project managers can easily forget about adapting security measures within this new approach (Daud, 2010). The spiral model uses iterative steps that can create changes in each phase of software development process. This will challenge security to ensure the application has few flaws or vulnerabilities that could be exploited. Failure to impose the right security measures at each of these phases can open up vulnerabilities for hackers to exploit and prove costly in patching.

This paper will explore areas where security can be engaged within each iteration step of the spiral development model. This way security can continue protection of the software product as it becomes ready for deployment. The paper is broken down into sections describing the spiral development model and what security methods are needed at each step. The first section describes an overview of spiral development model to set the framework of where security will be suggested. The next section describes security methods recommended at different parts of spiral development model. The paper will help organizations realize where to deploy security measures to reduce vulnerabilities in the software application.

Keywords: Spiral Development Life Cycle, Software Development, Risk

1. Introduction

Security measures have been well established within the waterfall development life cycle model. However, more software projects are using the spiral development life cycle model. Software developers and project managers can easily forget about adapting security measures within this new approach (Daud, 2010). The spiral model uses iterative steps that can create changes in each phase of software development process. This will challenge security to ensure the application has few flaws or vulnerabilities that could be exploited. Failure to impose the right security measures at each of these phases can open up vulnerabilities for hackers to exploit and prove costly in patching.

2. Overview of Spiral Development Life Cycle Model

Understanding the spiral development life cycle model for devising software applications is important to security professionals. Knowing what is done in each phase will help to plan what
specific security methods is needed. Unlike the waterfall model, the spiral model incorporates risk analysis with each iteration step of the model. This is the first time security is incorporated as part of the development model as compared to the waterfall model. Another feature of the spiral model is that development is done in small iterations over a period of time. Each iteration step will allow security measures to be constantly updated and applied. This is because requirements and software products can change with each step of the spiral model. Therefore, it is important for developers and security personnel to understand what is done in each phase in order to see where security measures need to be placed and updated.

2.1 Determine objectives, alternatives and constraints

The spiral model starts with a small amount of requirements and proceeds through each phase of the model (Purcell, n.d.). These are a partial list of the objectives, alternatives and constraints of the final model. This way the development process can start but with incomplete requirements. Additional functionality is added with each iteration step of the process until the product is completed (Purcell, n.d.). So in this phase a small amount of functionality is set, alternative implementation designs or methods are considered, and constraints on these alternative methods are looked at (Boehm, 1988). None of these are evaluated at this time in regards to risk. So at this time, security experts and developers will have limited information to consider security measures to use. The next step will evaluate the risks of each to help determine the best course of action.

2.2 Evaluate alternatives, identify and resolve risks

In this step, risks are identified and resolved. Risk management is used to perform this analysis. This step is used to keep the project on track as requirements may change over time (Purcell, n.d.). From the risk analysis a prototype can be developed. Each prototype is tested with the user to also help meet requirements. The process continues as requirements are more refined and alternatives need to be evaluated. If all risks are resolved, the process continues onto the next step in developing the next product level (Boehm, 1988). This can reduce security flaws and save costs before product implementation.

2.3 Develop, verify next-level product

The risk management results from the previous phase will determine how much time will be available for this phase (Boehm, 1988). In this phase, models are devised for future development work. These are adjusted during each step in the model to become benchmarks. From this software requirements are devised and validated to meet these requirements. Further iterations devise and validate the product design to make sure it meets the user needs. Lastly, the software product is tested in the last iteration of this phase. However, the next phase will devise the plans used to contribute to these parts of the development phase. Thus, each iteration step feeds information from one phase to another.

2.4 Plan next phases
The last phase involves developing plans to guide the next iteration. These include plans for requirements finalizing, development of the application design, and integration and test processes until the application is implemented (Boehm, 1988).

3. Implementing Security Practices

Implementing security practices in the spiral development model need to be flexible and stay focused through each iteration step of the model. That’s because security should be considered a high priority process (Kaur, Kaur & Singh, 2012). Secure software applications can be more resistant to attacks and failures. Thus, it is important for security practices to be implemented continuously within the spiral development model.

3.1 Security in the first phase

It is always best to involve security at the first stage of any software development process. This is a best practice according to Kissel, Stine, Scholl, Rossman, Fahlsing and Gulick (2008). Security professionals should be looking at privacy concerns, threats, and possible risks. A privacy assessment should be done to bring out concerns that need to be addressed. This is also a good time to devise threat models and revise them over time. The threat models can help identify where risks could appear. The model may have to be reviewed and updated as the spiral development model progresses through each iteration step. The model can help ensure that any risks are being addressed in further phases. While the waterfall model does this later in the design, doing a threat model early can reduce costs and number of risks found (Howard, n.d.). Security education of developers should be accomplished at this point. Part of this education is to provide secure coding guidelines for developers to follow. It is important that developers follow and use these guidelines continuously through the spiral development process according to Kaur, Kaur and Singh (2012). They will also be important when devising models and simulation later on. Still another security matter to consider in education is devising least privilege process for controlling access of users. This is a good time to begin devising the process so that security administration of application users can be established before the product is released. Establishing the process now will help reduce security risks in the next phase. According to Howard (n.d.), if developers are not familiar with security flaws or defects based on design, coding or testing, then they cannot devise secure software. Next, security architecture should be started to begin gaining an understanding of application. It can be adjusted as requirements become more defined. Thus, starting security here can result in reduced costs and more secure code as the development process moves forward.

3.2 Security in the second phase

According to Kaur, Kaur and Singh (2012), they recommend a security risk analysis be performed before the risk analysis done in this phase. They found that doing it before each iteration step of the cycle can identify security risks to the project that could create uncertainty in the final product. Security should also review each prototype just in case a security flaw does get embedded. Another security practice is to understand business tolerance to any risk and residual risk. This can help in how much security risk needs to be mitigated to appease users.
3.3 Security in the third phase

This phase requires a heavy security involvement throughout each iteration step. Also a formal review of the requirement specifications should be done to find errors. Davis (2012) sees this as a minimum requirement for devising secure software. From this, the security professional should build abuse cases to help illustrate the problems with the software design. They can identify where abnormal behaviors can happen (Daud, 2010). All of these security methods will be important when the coding and testing begins (Kaur, Kaur & Singh, 2012). To prevent possible exploits such as SQL injections, the inputs and outputs should be checked to make sure no unexpected actions could affect the performance of the software application. Fuzzing testing is a good choice of input validation testing to use. Fuzzing testing can find how invalid or random data can affect the software application (Kaur, Kaur & Singh, 2012). More testing such as white box, black box and penetration testing should be done to shake out unknown security issues that may not appear until after production. Together, these tests can find security flaws and vulnerabilities before product release and correct them to prevent possible exploitation by hackers.

3.4 Security in the fourth phase

In this phase security should ensure that any plans developed during this time document security practices from previous phases. This way they will not be loss or forgotten as different iterations of the model are implemented. Specifically, areas of disaster recovery, configuration management (CM) and quality assurance (QA) should be established and documented. Kissel, Stine, Scholl, Rossman, Fahlsing and Gulick (2008) showed these document being developed later in the waterfall model. However, doing them earlier within the Spiral model can help implement the QA and CM during the process through the spiral model. The benefit is that they can reduce risks and costs within the development process.

4. Conclusion

Preventing attackers from exploiting software flaws is important to devising applications. This requires security professionals to be engaged at the start of the spiral development model and each iteration step of the model. This challenges them to be adaptable to changes in the requirements and development of the software application. The practices recommended in this paper helps meet this challenge. Incorporating security measures in all iterations of the spiral development life cycle model can save costs and prevent future vulnerabilities. By following some simple security practices, software developers and security professionals can also devise secure software for their users (Davis, 2012). Also with each iteration, developers gain experience in applying security when coding and testing software products. Otherwise, they risk software flaws that could contribute to the loss of a firm’s data.
References


