International Journal of Computer & Information Technologies (IJOCIT)
Corresponding Author: *Nasser Modiri*
April, 2015

Developing a New Approach to the Architecture, Design and Secure Implementation of Web Applications

Afsaneh Gilani[1], Nasser Modiri[2*], Alireza Nikravanshalmani[3]
Department of computer, Karaj Branch, Islamic Azad University, Alborz[1,3]
Department of Electrical, Computer & IT, Zanjan Branch, Islamic Azad University, Zanjan[2]
Iran[1,2,3]
Email : nassermodiri@yahoo.com

Abstract: One of the main challenges of web-based applications is their security vulnerabilities, which are caused by the lack of concern for security issues in the lifecycle of software development. Therefore, concern for security requirements in the lifecycle of software development is one of the essential components of secure software development. Most software troubles are caused by the inappropriate design and development of the software. Almost 50% of security holes emerge in the software design phase. In this paper, the most important secure methods for software development and the characteristics of each method were discussed. Afterwards, using the discussed methods an approach was proposed based on the security requirements engineering for the development of secure web applications.

Keywords: Software security, Security Requirements Engineering, Secure Software Life Cycle, Secure Software Architecture and Design, Secure Software Implementation.

1.Introduction

Today, web-based applications are of significant importance because of the storage of high volume of information as well as processes that must remain secure. Therefore, it is important to ensure the system is developed based on user requirements while ensuring the security of software systems, which is equally important. [1]

One of the reasons for the success of most of the attacks to software systems is the vulnerability of such systems. In order to develop secure software systems, it is necessary to provide a secure environment and consider security in the primary design of the system. Hence, the software lifecycle (including the design and architecture software) security requirements must be taken into account. [2]

At best, the security in the system coding phase is to be considered primarily because organizations more care about the function than software security. Moreover, security experts rarely take part in the development process. Hence, systems are not developed securely and as a result software systems have security flaws. It is economic to consider security in the early stages of system development as it will also result in a more powerful design [3].

Security requirements engineering (SRE)[4] is a new field of software development, which focuses on the incorporation of software security in the early stages of the design of software systems. The lack of concern of developers for SRE leads to the development of security flaws in software because the major objective of system developers is to build an application system instead of a secure application system. In general, software flaws are of the following two origins: [5]

- Problems caused by bad design decisions: These defects are usually known as flaws.
- Coding errors which are known as bugs

According to McGraw's theory, 50% of software security problems originate from the design phase. Design flaws are not identifiable in the coding phase. The main goal of attackers is to gain unauthorized access to vital business and private information. Each system should be architected such that it can protect the information against any sort of attack[6]. Moreover, according to the OWASP (Open Web Application Security Project) theory, software design level flaws are less investigated while they pose the highest risk to the software. It is hard to detect such flaws through static or dynamic application scans and detection of such flaws calls for a profound understanding of software architecture. Software vulnerability can occur in each of the software lifecycle phases but the design level is the phase with the most susceptibility to security vulnerabilities.[7]Although the design level is highly important for the development of secure systems, the implementation and coding phase also play an important role in the development of secure software.

Moreover, although there is a great deal of information about the development of secure software, there are still questions about development of such software. Because of the smart attacks aimed at web applications, the security of web applications has become a major challenge in the field of software development. Hence, the main question or problem is how to overcome software security problems. In this study, the most important secure software methods were examined and were used as the basis for a new approach to the architecture, design and secure implementation of software. Different parts of this article are as follows: Section two discusses related work and section three investigates the software lifecycle. The third section includes security activities in the architecture level while section four discusses the security measures proposed for the detailed design level. Section five discusses security measures for the coding phase and finally provides a conclusion.

2.Literature Review

The MS-SDL (Microsoft Security Development Lifecycle) method, which was introduced by Microsoft, particularly values the training and awareness of users. In this method, threat modeling is used to identify security vulnerabilities in the design and architecture phases[8].

CLASP (Comprehensive Lightweight Application Security Process) was developed by OWASP in 2006 and is comprised of 30 security activities. The roles defined in CLASP include the roles of manager, architect, requirements specifier, designer, implementer, tester, and security auditor.CLASPincludes templates,checklists,and guidelines to support different activities [9].

In the Touchpoints method a set of industrial experiences are included as best practices. Viewpoints are also a combination of black-hat (destructive) and white-hat (constructive) activities. Both groups of activities are necessary for achieving effective security results.In this method, the three major principles of software security include risk management, software security viewpoints, and knowledge [10].

The following table shows activities proposed for the secure software development methods in each phase of the software development lifecycle.

At the architecture level, the proposed SDL technique aims at identifying the security vulnerabilities of threat modeling. In the detailed design phase, this technique focuses on reduced attack surface. For example, the proposed technique is used to reduce attack surface, remove unnecessary features, and limit privileges.

Similar to SDL, CLASP also supports threat modeling in the architecture phase. At the detailed design level, CLASP acts as a supplement to SDL because of the attack surface reduction activities it supports. In fact, SDL is mostly focused on limiting privileges whereas CLASP is focused on limiting the access points.

The main focus of the Touchpoints method in the design phase is on risk analysis. In the design and architecture phases security principles, such as principle of least privilege shall be taken into account. Moreover, in these phases the attacks should also be documented.

In the design phase, the most important risks and risk-coping mechanisms are identified. Although smart design is a good start point, it is not enough for software presentation. The implementation part has also its specific tricks [11]. The SDL and Touchpoints methods propose secure coding principles for the implementation phase. In this paper, by combining the strengths of secure software development methods an enhanced approach to secure software development is developed [12].

Table 1: Activities proposed for the secure software development methods

Software lifecycle phases	SDL	CLASP	TouchPoints
Requirement	Identifying security requirements	Using misuse cases to determine security requirements	Determining resources
Design	Threat modeling, attack surface analysis, use of security approaches for secure software development	Threat modeling, attack surface detection, use of security guidelines, considering security issues in the class diagram	Risk analysis
Implementation	Applying secure coding standards, using static code analysis tools, and testing	Applying secure coding instructions	×
Software test	Code analysis and use of secure testing tools	Using software security test tools and presenting security checklists, Penetration Testing	Black box test, white box test, Penetration Testing

3. Software Lifecycle

In general, the software lifecycle consists of the following six phases: requirements analysis, design, coding, testing and support. The ISO/IEC 12207 standard divides the software design phase into two parts: software architecture and detailed design. According to this standard, in the software architecture phase the overall high-level structure and components of the software are defined. However, in the detailed design phase each component is examined and details of the design of each component are also determined.

4. Security Activities Proposed for the Software Architecture Phase

The first step in the design level is to determine the software architecture. This phase is known as the top-level design phase, in which system components are designed based on the identified requirements. Software architecture is usually shown by physical and logical diagrams. In software architecture elements represent the system performance. Activities proposed for the software architecture phase are shown in figure 1.

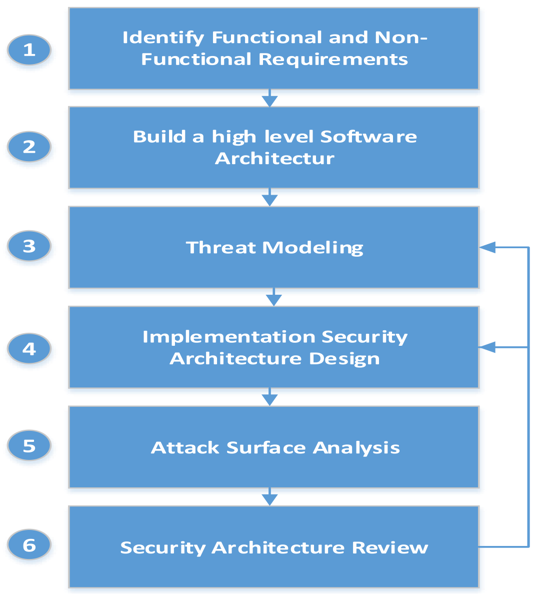
1
Identify Functional and Non-Functional Requirements
2
Build a high level Software Architectur
3
Threat Modeling
4
Implementation Security Architecture Design
5
Attack Surface Analysis
6
Security Architecture Review

Figure 1: Security activities for the software architecture phase

Each of the activities shown in the figure 1 is described below.

Stage one: Identifying functional and non-functional software requirements

In this phase, the functions required by the system as well as the security goals of the software (which are determined in the needs assessment phase) are examined. The most important security goals include confidentiality, integration, availability, and data integrity.

Stage two: Overall software architecture design

In this phase, all of the components required for the architecture shall be identified. Inputs of this phase include use case diagrams, functional requirements, non-functional requirements (efficiency, reliability,andsecurity), required technologies, and deployment environment.

Stage three: Threat modeling

Threat modeling is an effective approach to the detection of flaws at the architecture level. The objective of this phase is to assess threats associated with system assets and present acceptable security controls. In this phase, all of the threats, vulnerabilities and security issues associated with each component are identified.

Stage four: Implementation of secure architecture design

After identifying the threats to software architecture, it is necessary to examine and study techniques for reducing the risks of software architecture. In this phase, components necessary for ensuring security should be identified.

Stage five: Attack surface analysis

Attack surface analysis includes identification of the system parts that call for security tests and assessments for the detection of vulnerabilities. The attack surface is defined by architects but the developers and testers should also examine attack surfaces.

Stage six: Review of software architecture design

In this phase, documents prepared at the architecture level are reviewed. The resulting document is reviewed with an aim to consider the security goals defined in the software architecture phase. In this phase, the proposed architecture should be examined using OWASP ASVS [13].

The outputs of the secure architecture design include the following:

- Overall software diagram
- A list of software architecture threats

5. The Proposed Approach for the Detailed Design Phase

Detailed software design follows the architecture design phase. In this phase, the details of components and interfaces defined by software architecture are examined. Details are documented such that they are comprehensible for the programmer in the implementation phase. Activities proposed for the detailed design phase are illustrated in figure 2.

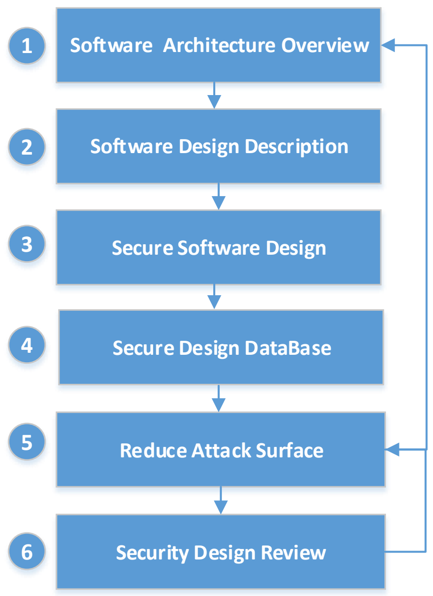

Figure 2: Security activities for the detailed design phase

Each of the above activities is described below.

Stage one: An overview of software architecture

In this phase, components determined in the software architecture as well as the interactions between the components are studied. Moreover, results of threat modeling and the threat model corresponding to the software architecture are analyzed.

Stage two: Software design description

At this stage the details of components specified in software architecture are presented in a comprehensible manner to the programmer. In addition, component interfaces, components design, workflow and algorithms should be defined in this phase.

Stage three: Secure software design

For the purpose of secure software design it is necessary to employ secure software design guidelines and adopt security approaches to secure software design. Moreover, it is recommended to use security modeling languages such as UMLsec[14] in this phase.

Stage four: Secure database design

One of the essential aspects of software security is database security. The most important techniques proposed for the security of databases include the encryption and access control methods.

Stage five: Reduce attack surface

The attack surface of software products includes a part of the code, interfaces, services and protocols available to all users (especially to non-reliable users). The process of surface attack analysis examines all of the interfaces, protocols and executable codes. This process reduces the access of unauthorized users to executable codes.

Stage six: Software design review

In this stage, the detailed design documents are examined using checklists so as to determine the security vulnerabilities.

The outputs of this stage include the following:

- Components description
- Design of modules embedded in layers or subsystems

6.The Proposed Method in the Implementation Phase

In the implementation phase, developers implement software modules based on the design documents, which determine the tasks of components and the interactions between them. Next, modules are merged to obtain an integrated system. The implementation phase is important for successful development of secure software. Security bugs in the implementation phase are introduced into the system as a result of poor coding, the lack of security knowledge and the use of non-secure methods. In the implementation phase, developers use the documents resulted from the detailed design level.

Activities that should be conducted in this phase include the following:

Stage one: Review of the detailed design document

In the implementation phase, the outputs of the detailed design level are usually used to present the software package. Results of threat modeling are also used as important guidelines in the implementation phase. At this level, developers pay special attention to the accuracy of codes so as to reduce vulnerabilities.

Stage two: Secure software coding

In this stage a suitable language should be used for writing codes. Moreover, coding should be carried out based on programming instructions and secure coding standards. The most important points about secure programming, which were introduced by CERT, include the following: input validation; considering compilers' warnings; a review of the architecture and design to identify security policies; simplicity of implementation; the principle of minimum access; in-depth defense mechanism; and application of secure coding standards.[15]

Stage three: Code analysis

In order to analyze the codes, it is necessary to use static and dynamic code analysis methods. The static analysis method is used to review security vulnerabilities when coding is finished. However, dynamic analysis is the process of run time software testing. In this process, the attacker's behavior is analyzed and vulnerabilities are identified.

Stage four: Code review

In this stage codes are assessed and examined using secure coding checklists based on the threat model so as to identify the vulnerabilities. Using checklists, it is possible to study the most important security vulnerabilities such as authentication, authorization, access control, input validation and output validation.

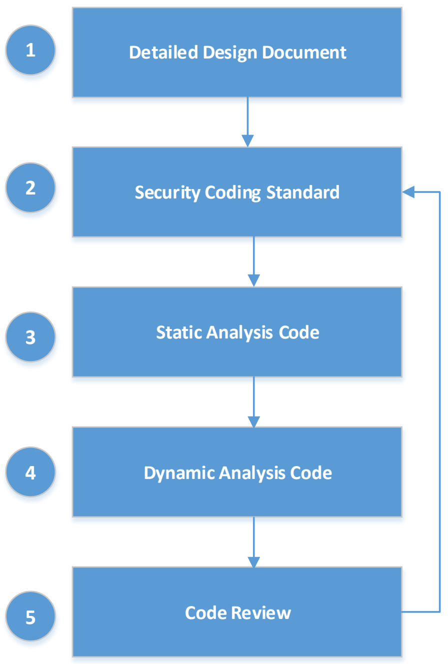

Figure 3: Security activities in the coding phase

7. Conclusion

For the purpose of secure software development it is necessary to consider security requirements throughout the software lifecycle. Among all of the different phases of software lifecycle, the design and implementation phase plays the key role in the development of secure software. Every chance of secure software development should be used to determine the security activities for each of the development stages (i.e. needs assessment, design, implementation and testing).

The characteristics of secure software development methods based on the parameters for each of the software development stages include the following. In the needs assessment phase the software security requirements are identified. At the design level, the threat modeling technique, secure design principles and instructions, and secure design templates are employed. In the implementation stage a secure programming language is used to reduce security risks and errors. In this phase, it is also necessary to follow secure coding standards and instructions. Attack surface analysis, as an important parameter in the secure software lifecycle, should be considered and the attack surface should be reduced to the possible extent.

References

[1] Salini, S. Kanmani,(2013)," Model Oriented Security Requirements Engineering(MOSRE) Framework for Web Applications", Springer, pp. 341-353.
[2] Ramin Shirvani, Nasser Modiri,(2011)," Software Architectural Considerations For The Development of Secure Software Systems", IEEE, pp. 84-85.
[3] Daniel Mellado et al,(2010)," A systematic review of security requirements engineering", Elsevier, pp. 153–165.
[4] P. Salini, S. Kanmani,(2012),"Survey and analysis on Security Requirements Engineering", Elsevier, pp. 1785–1797.
[5] Christoph Hochreiner et al,(2014),"Using Model Driven Security Approaches in Web Application Development", Springer, pp. 419–431.
[6] Asoke K. Talukder , Manish Chaitanya,(2009)," Architecting secure software systems", Taylor & Francis Group.
[7] S. Rehman,K. Mustafa,(2009) , "Research on Software Design Level Security Vulnerabilities", ACM.
[8] Michael Howard, Steve Lipner,(2006),"The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software", Microsoft Press.
[9] OWASP CLASP,(2015),"OWASP CLAPS Project " . [Online] . Available:https://www.owasp.org/index.php/CLASP.
[10] McGraw,(2006)," Software Security: Building Security In", Addison Wesley.
[11] AXELLE APVRILLE , MAKAN POURZANDI,(2005)," Secure Software Development by Example",IEEE,pp.10-17.
[12] Bart De Win et al,(2009),"On the secure software development process:CLASP,SDL and Touchpoints compared", Elsevier, pp. 1152–1171.
13.OWASP,(2014) ,"Application Security Verification Standard " , [Online] . Available:http://www.OWASP.org.
[14]Jan J¨urjens,(2002), " UMLsec: Extending UML for Secure Systems Development " , Springer,pp.412–425.
[15] Robert Seacord,(2011)," Top 10 Secure Coding Practices " , [Online] . Available:
https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices.