International Journal of Computer and Communication Technology

Volume 8 | Issue 1 Article 12

January 2017

Secure Web System Development

Ankush P. Deshmukh
Department of Computer Technology, VJTI, Matunga, Mumbai, India, ankush25d@gmail.com

Vaibhav G. Korat
Department of Computer Technology, VJTI, Matunga, Mumbai, India, vaibhavkorat@in.com

B. B. Meshram
Department of Computer Technology, VJTI, Matunga, Mumbai, India, bbmeshram@vjti.org.in

Follow this and additional works at: https://www.interscience.in/ijcct

Recommended Citation
Deshmukh, Ankush P.; Korat, Vaibhav G.; and Meshram, B. B. (2017) "Secure Web System Development," *International Journal of Computer and Communication Technology*: Vol. 8 : Iss. 1 , Article 12.
Available at: https://www.interscience.in/ijcct/vol8/iss1/12

This Article is brought to you for free and open access by Interscience Research Network. It has been accepted for inclusion in International Journal of Computer and Communication Technology by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

Secure Web System Development

Ankush P. Deshmukh, Vaibhav G. Korat & B. B. Meshram

Department of Computer Technology, VJTI, Matunga, Mumbai, India
E-mail : ankush25d@gmail.com, vaibhavkorat@in.com, bbmeshram@vjti.org.in

Abstract **-** The recent development in the field of Web system technology has transformed the software industry radically by integrating a wide range of web users, vendors, and enterprise applications worldwide. In Web-based system, a security requirement is a critical issue. This difficulty is due to the complexity level of such systems as well as their variety and increasing distribution. The World Wide Web has become a highly adopted platform for web system. In order to avoid the high impacts of software vulnerabilities, it is necessary to specify security requirements early in the development on a detailed level and it needs to be built into the application design up-front by explicitly stating the security approach. A current web system faces major security problems because security design is not integrated into the Web Engineering Development Process. Due to insufficient support for a concrete and assessable level the application security and the software security is invaded. This paper emphasizes on need of security at Development Lifecycle (SDLC) of web system and integration of security in web system development life cycle.

Keywords **-** *Secure Web System (SWS), Secure Development Life Cycle (SDLC).*

I. INTRODUCTION

Security is an essential quality aspect of software products and in some cases; it is the most important non-functional requirement of the system. Using software systems in critical environments and relying on its functionality may lead to huge amount of risks and damages if security weaknesses are present. The need to be more careful about system security and information implies the need to have a life cycle of software development activities focused on specific security as a means to achieve greater assurance of integrity, reliability and availability of information. Solving security issues at the end of the QA (quality assessment) period isn't the best way and is often introduced towards the end of the project. Forward Developers and architects need to design and understand secure coding techniques and how attackers exploit applications. Although there are a number of great tools available to help automate security of web applications, no tool or suite of tools can be effective without the correct processes in place and educated, informed people creating and testing the web apps.

Web applications are the front-ends to most business systems today. Web application system serves a multitude of disparate functions within a complicated mix of architectures. Nowadays, the Web is extensively used as a major means of communication with the external world as well as means of communication within an organization. It is also as a tool to assist in carrying out its business process in a more effective way. Hence Web applications, which have been specifically designed for web-based environment, have received a lot of attention in the IT industry and lot of development is taking place in the field of Web system and services technology. More and more sensitive data is moving onto the web which brings new application security and information confidentiality challenges. Among all vulnerabilities, 92% of reported vulnerabilities are in applications. So, security plays crucial role in web system and services. Also Firewalls, intrusion detection and antivirus systems simply cannot solve secure software problem [5].

Effective security requires not only prioritizing security at the beginning of a project, but also increasing its visibility throughout the lifecycle of the project. Software grows up through its life cycle, so software development methodologies should pay special attention to security aspects of the product. Secure Web Engineering (SSE) is about building web application that can withstand attacks proactively to preserve fundamental security properties, namely, confidentiality, integrity and availability of critical assets. Current most software security often fails since its development is generally based on ad-hoc foundations. There are increased risks of security vulnerabilities that are introduced into software in various stages of development. So, to avoid inconsistencies and ambiguities, it is essential that the system is developed in a secure fashion from the beginning. By considering various strength and weaknesses of various engineering techniques, paper contains the formulated framework by

integrating, investigating various web analysis and design methods.

II. GUIDELINE BEFORE GOING FOR SDLC

To design and develop sustainable Web systems is the challenge for better

- usability, interface design, and navigation
- comprehension
- performance (responsiveness)
- security and integrity
- evolution, growth, and maintainability
- testability
- mobility

To understand the need for security, all members of the team is to gather answers of questions like: [6]

- *What are the common security requirements for web applications?*
- *What are the vulnerabilities found on each requirement?*
- *What are the solution methods available?*
- *What action to mitigate?*
- *What is the language to be used?*
- *What are the impacts of the architectural choices, frameworks and components?*

By the findings, everyone knows the correct way to avoid vulnerabilities besides what is the best architecture or pattern to be used and understand what has to be tested on each requirement.

The guide can be written and updated by all members of the team in order to create a mechanism of information exchange and dispersion.

a) *Requirement Analyst*: helps in the writing of the requirements in a measurably way.

b) *Test Engineers***:** responsible for specifying vulnerabilities and how to test each one of them.

c) *Architect***:** proposes solutions and architectural patterns for the vulnerabilities and issues found.

d) *Developers***:** proposes technologies solutions and languages that can be used to mitigate the risks analysed by the impacts of the vulnerabilities. Finally the guide must be validated by a security specialist.

Essential security criteria for Web System Development identifies are as follows [4]:

i. Active organizational support for security in the Web development process which includes the managerial support for security integration into the development process, support for developmental staff, end-user communication and security education.

ii. Proper Controls in the development environment which encompasses policies, knowledge, technology, and processes. These controls help to provide structure to the development environment.

iii. Security visibility throughout all areas of the development process which implies that the development process needs to be security focused. The term security focused translates into the use of effective and efficient designs, standards and procedures, while implementing good coding practices, addressing security issues having specific security testing criteria, and acquiring feedback from the end-user that is security specific.

iv. The security requirements of the business need to be identified as early in the development process. So, it contains delivery of a cohesive system, integrating business requirements, software and security.

v. Prompt, rigorous testing and evaluation include as much testing as possible from the design and programming perspectives that have automated and manual scripts, code reviews, and black, white and grey box testing.

vi. Trust and Accountability: encompasses the management of risk, the implementation of appropriate controls, the education of employees, effectiveness monitoring and actions performed in the application.

III. SECURING THE WEB SYSTEM DEVELOPMENT LIFE CYCLE

Each phase plays a role in the quality of the overall security of final product and therefore, must be considered from a security perspective.

a) Requirements and Use cases

b) Architecture and Design

c) Code Implementation and Build

d) QA/Testing

e) Deployment and operation

f) Deliver

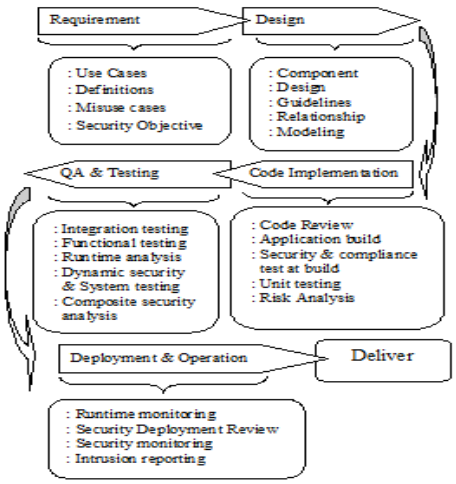

Fig.1: Securing Web system Throughout the Lifecycle

A. Requirements and Use Cases:

It has the activities to derive complete functional and security requirements. The behaviour and deployment environment of the software should be specified using use cases along with detailed functional requirements (diagrams/text). The requirements specifications ought to be inspected (multiple times, if required) for identifying and removing software errors. Abuse cases should be specified. Information about software assets, attackers, attacker's interests, attacker's resources, attack surfaces, and threats to existing software of the same domain must be collected. Risk analysis should be performed on the identified threats to calculate the collective potential damage that each of them can cause to different assets. This information can be used to prioritize threats [9].

High-level security requirements such as preservation of confidentiality, integrity, and availability should be specified to mitigate identified threats. High-level security requirements can be prioritized by performing a cost benefit analysis. Security mechanism with higher priority should be implemented first if there are budgetary constraints. System security errors and already specified low-level security requirements may be identified through inspections. [7].

Steps for integrating security into the requirements phase:

- Discuss and define security requirements based on corporate policy, compliance and regulatory mandates.
- Security and audit teams should assess business requirements and functions of the web application and begin to formulate mis-use cases for use during testing and acceptance.

Integrating security at this phase, will significantly reduction in time-to-deployment and reduction in acceptance bottlenecks.

B. Architecture and Design:

Design is the representation of decisions taken to fulfil requirements. Design specifies two aspects of the system:

- Static structure
- Dynamic behaviour

Detailed functional design (including design of security mechanisms) should be specified in a secure manner by secure design patterns and secure design guidelines. The design must be inspected (multiple times, if required) for identifying and removing software errors. The threat model developed should be enhanced. Risk analysis should be performed on the identified threats to calculate the potential damage that each of the threats can cause.

Software security errors and previously specified secure design decisions should be identified through inspections. Detailed design specifications are created, that show developers exactly which security controls must be included and how the components will interact with the overall web ecosystem.

Steps for integrating security into the architecture and design phase:

- Perform risk assessment in context of the application's proposed architecture and deployment environment; and use the results to supplement the baseline security controls.
- Assess security implications of interaction with legacy systems and security implications of data "flow" between components, tiers, or systems.
- Secure design decisions to remove threats can be prioritized based on a cost/benefit analysis.
- Document any context-specific exposures (i.e., vulnerabilities that are dependent on how and where the application is deployed) that need to be addressed during implementation/rollout.
- Consider dependencies and exposures created by interactions with mash-ups, SOA, and partner services.

C. Code Implementation and Build:

Code Implementation/Assessment is the third phase of the SDLC. The system will be installed and evaluated in the organization's operational environment. In this phase, re-usable policies increase accuracy of risk analysis and cleaner/less vulnerable code is delivered to QA.

Automated static code tools that are integrated into the IDE provide developers with checks and guidance as code is written and before check-in. Automated tools can also be used during build to check the code against policy templates for compliance and for a deeper look at code level security issues. Fewer coding errors/vulnerabilities discovered during testing resulting in faster deployment cycles [9], [2].

Steps for integrating security into the code implementation and build phase:

- Install automated static source code checking tools that are integrated with developer IDEs
- Developers can also perform automated code reviews with stand-alone coding tools before check-in.
- Using automated or manual code reviews, security and audit teams spot check code modules for compliance conformance and security risk prior to build.
- To check for security exposures and policy compliance, implement automated static code scanning during the build process.
- Track developer coding errors with the use of tools and provide explanatory feedback on security risks introduced and reasons behind it.

D. Quality Assurance / Testing:

Security specific tools for testing applications range from one-off stand-alone solutions and services that assess the completed application to fully integrated suites. Integrated solutions provide multi-phase support towards a repeatable web application security lifecycle.

Integrated suites can be implemented at multiple points in the process and provide metrics and feedback for on-going continuous improvement. Phase benefits to better communication between application stakeholders and improve bug fix and release cycles [7].

Key indicators:

- Solutions that can be applied at multiple points throughout the process (education, implementation, testing, deployment)
- Integration with existing SDLC tools such as build and QA solutions, Easy to understand, interpret, and use results
- Comprehensive reporting for compliance
- Continuous improvement support – identification of developers or application types that require additional training or security controls

Steps for integrating and improving security in the QA/testing phase:

- Validate test findings in a production architecture that includes existing compensating controls such as firewalls and IPS
- Finding out the problems and errors which is grouped into three categories:
  1. Insecure interaction between components
  2. Risky resource management
  3. Porous defences
- Prioritize discovered vulnerabilities based on both security and business needs

Deliver fix recommendations to development with specificity to line of code or dependent API, service, or library.

Analysis Tools for analysis phase:

Security testing solution is a part of software development life cycle. The tools used in the software development life cycle for analysing the security of Web applications can be roughly divided into three different types of tools: *[7]*

- White box analysis tools
- Black box analysis tools
- Gray box analysis tools

The test called as Penetration test goes a step further and validates whether attack pathways result in risk to the organization and attempts to penetrate own applications. The key value of a penetration test is that, it bridges the gap between the discovered vulnerability and the exploitable asset so to make an informed risk decision [8].

a) White box analysis:

White box analysis has the highest access to this information and can be seen as approaching security from the developer's perspective. White box analysis

can quickly determine the complete attack surface and create the necessary set of tests.

White box analysis includes three primary techniques:

- *Architectural analysis*: It is often referred to as threat\modeling, seeks to enumerate the goals an attacker has within the software and introduces countermeasures for each of these threats.
- *Source code analysis***:** This scans the source code and traces user input through the application to find vulnerabilities and bad coding practices.
- *Static binary analysis***:** which operates similar to source code analysis but at the binary level, which allows it to find contextual risks and platform-specific problems.

Challenges in White box analysis:

- Poor coding practices can falsely be detected as vulnerabilities.
- Software cannot be easily tested remotely in a distributed environment.
- Access to the design specifications and source code is not always possible

b) Black box analysis:

Black box analysis starts from the perspective of an attacker with no access to any of this information. Black box analysis involves examining the software or system with no prior knowledge of the environment. As it more accurately reflects the immediate risk posed by the outsider, risk findings from this analysis are often prioritized.

Black box analysis includes two primary techniques:

- *Vulnerability scanning***:** It is realized by using a large database of known vulnerabilities and trying to identify a known vulnerability in an application. This can be either passive or active. This scanning only tests for known vulnerabilities.
- *Dynamic analysis***:** It tries to automatically scan and document the attack surface, test the application by means of fault injection, and determine the presence of a vulnerability based on the response. Dynamic analysis can discover unknown vulnerabilities

Challenges in Black box analysis:

- It is impossible to determine code coverage; obscured or hidden functionality can be missed.
- Logical design flaws cannot be detected easily.

c) Gray box analysis:

Gray box analysis combines both black box and white box analysis in a powerful way to improve accuracy and coverage. We can get the benefits of both approaches with minimizing the potential that important issues are missed.

Challenges in Gray box analysis:

- It requires correlating the result sets that are obtained in different phases of the software development life cycle.
- Success rate is high, when gray box analysis becomes a fundamental part of the software development life cycle, allowing for smooth integration of different test results obtained throughout the life cycle.

E. Deployment/Production:

Once the web application is live in production, additional testing and monitoring can be implemented to ensure data and services are protected.

Automated security monitoring of production web application system provides assurance that the web system is performing as expected and not exposing information or introducing risk. This phase gives better integration between dynamic testing and production application controls such as web system firewalls or IPS emerging compliance requirements before code re-write. Web system (application) should be monitor on a 24/7 basis. Also use the feedback loop for continuous improvement of web system.

Steps for integrating and improving security in the deployment/production phase:

- Plan and conduct system certification activities in synchronization with testing of security controls.
- Monitor mis-use to affirm vulnerabilities which were not exploitable in testing are not exploitable in production.
- Get information by comparing residual risk assessments before deployment with the production exposure areas.
- Provide feedback to testing team from above information.
- Monitor data leakage i.e. places and fields where it is used, sent or stored inappropriately.
- Identify, in terms of security, the weaknesses and strengths of the programming language.

- Implement web system firewall, IPS, or other compensating control to mitigate exposures before code fix or in response to new regulations.

More on analysis; every phase has its own features and functionality. But, analysis is the activity which can be carried out on every phase of development lifecycle. Here we have suggested analysis approach for giving better strength to final product.

- *Architectural security analysis* should be a part of your design phase.
- *Source code analysis* should be integrated into your development phase.
- *Dynamic analysis* starts in the development phase and lasts all the way to your delivery phase.
- *Binary analysis* is something that you typically do once during the audit part of the delivery phase for each new build of your software.
- *Vulnerability scanning* is a repeating task that should be do ne periodically as long as your software is operational.

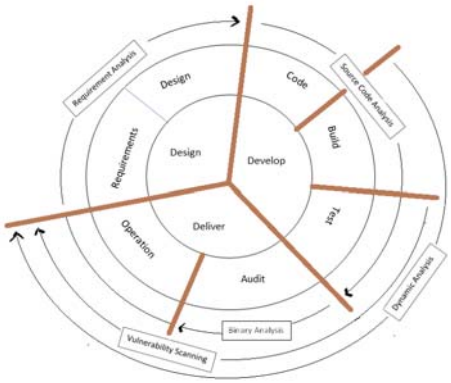

Fig. 2 : Different Analysis throughout the cycle

IV. DEFINING SECURITY CLEARLY

It is important to know exactly what the difference between security requirement and security features.

Security requirements: are tasks that must be done.

- e.g. all passwords must be stored as encrypted.

Security features: are functionalities that must be available.

- e.g. account management must be possible through a Web interface.

The following table summarizes some of the strategies for building up secure web development applications.

a) *Input Validation*: Do not trust on customer-side confirmation. Accept centralized input confirmation only.

b) *Authentication*: Employ strong passwords and provide password accounts disablement and their period of expiration.

c) *Authorization*: Accept approval granularity, implement partition of privileges, and employ least privileged process and accounts.

d) *Configuration Management*: Employ concrete authorization and authentication on management interfaces.

e) *Sensitive Data*: Protect the communication course and offer concrete accesses on sensitive encrypt data records.

f) *Session Management*: Encrypt the data of verification cookies. Protect the course and session status from unofficial accesses.

g) *Cryptography*: Employ proper key size and algorithm. Employ tested and attempted platform attributes.

h) *Parameter Manipulation*: Do not believe those fields which are easily manipulated by customer such as, query strings, HTTP headers, and etc.

i) *Exception Management*: Do not disclose responsive particulars of application. Employ structured exception handling.

j) *Auditing and Logging*: Daily analyze and protected access to log files and back up. Log activity and audit throughout the entire application tiers.

Sometimes there will be a tight deadline for the code to be rolled out to production and even the code reviews may miss a seemingly unobvious error. Such subtle errors may lead to significant security bugs in the overall system. So, Apart from penetration testing different tools should be used to check the vulnerabilities in coding.

- Pscan
- Flawfinder
- RATS (Rough Auditing Tool for Security)
- Splint (Secure Programming Lint)
- ESC/Java (Extended Static Checking for Java)

– MOPS (MOdelchecking Programs for Security properties)

V. CONCLUSION

The lack of security discussion in the beginning of the development process, lack of encouragement for re-usable components, lack of follow up after design approval, and lack of employee understanding of the role security plays in the web system development process. When looking at the broader picture of the development life cycle, it is found that, each of the security analysis techniques have their function in the software development life cycle. Although a thorough security audit during the delivery phase is an important aspect in securing your software properly but, it is vital to develop a security process that addresses security issues throughout that entire process. Security is an important quality aspect of web systems and to achieve this goal, we should attend to it during development life cycle. For complete coverage, combine white box and black box testing must be combined into one security testing solution as part of the development life cycle. whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

Figure above shows how architectural analysis, source code analysis, dynamic analysis, binary analysis, and vulnerability scanning can be used and combined throughout the design, development, and operational phases, security must be integrated into every step of the development life cycle.

Finally vulnerability scanning is a repeating task that should be done periodically as long as your web system is operational. Then we are looking forward to security modeling in designing web system.

REFERENCES

[1] S.W. Smith, CARLISLE ADAMS, Building Secure Web-Based Environment, PUBLISHED BY THE IEEE COMPUTER SOCIETY, 1540-7993/05, IEEE SECURITY & PRIVACY @ 2005 IEEE.

[2] Muhammad Umair Ahmed Khan and Mohammad Zulkernine, Activity and Artifact Views of a Secure Software Development Process, IEEE International Conference on Computational Science and Engineering,2006

[3] Noopur Davis and Watts Humphrey, Samuel T. Redwine JR., Gerlinde Zibulski, Gary Mcgraw, Processes for Producing Secure Software, IEEE COMPUTER SOCIETY, 1540-7993/04/$20.00 © 2004 IEEE.

[4] William Bradley Glisson, Andrew McDonald, Ray Welland, Web Engineering Security: A Practitioners Perspective, ICWE'06, July 11-14, 2006, Palo Alto, California, USA, ACM 1-59593-352-2/06/0007.

[5] Suhair Hafez Amer', Major Jeffrey W. Humphries', Ph.D. and John A. Hamilton, Jr.', Ph.D., Senior Member, ZEEE , Survey: Security in the System Development Life Cycle, Proceedings of the 2005 IEEE Workshop on lnfomation Assurance and Security United States Military Academy, West Point, NY.

[6] Rodrigo Elia Assad, Tarciana Katter, Felipe Silva Ferraz, Leopoldo Pires Ferreira, Silvio Romeiro Lemos Meira, Security Quality Assurance on Web-based Application Through Security Requirements Tests, 2010 Fifth International Conference on Software Engineering Advances.

[7] IBM Rational AppScan, Practical Approaches for Securing Web Applications across the Software Delivery Lifecycle.

[8] Gary McGraw, Bridging the Gap between Software Development and Information Security, IEEE COMPUTER SOCIETY, SEPTEMBER/OCTOBER 2005.

[9] Wei Huang, Ru Li, Carsten Maple, Hongji Yang, David Foskett, Vince Cleaver, Web Application Development Lifecycle for Small Medium-sized Enterprises, The Eighth International Conference on Quality Software.

[10] Richard Kissel, Kevin Stine, Matthew Scholl, Hart Rossman, Jim Fahlsing, Jessica Gulick, Security Considerations in the System Development Life Cycle, NIST Special Publication 800-64.

[11] William Bradley Glisson and Ray Welland, Web Engineering Security (WES) Process, University of Glasgow 2006, Department of Computing Science Technical Report - TR-2007-243.

[12] PDF: Improving Your Web Application Software Development Life Cycle's Security Posture with IBM Rational AppScan.D. Kornack and P. Rakic, "Cell Proliferation without Neurogenesis in Adult Primate Neocortex," Science, vol. 294, Dec. 2001, pp. 2127-2130, doi:10.1126/science. 1065467.

❑❑❑