International Journal of Mathematical Archive-2(11), 2011, Page: 2491-2496
Available online through www.ijma.info ISSN 2229 – 5046

SECURING WEB APPLICATIONS BEYOND SDLC

*B. V. S. S. R. S. SASTRY[1] & K. AKSHITHA[2]

[1]*Department of Information Technology, Aurora's Engineering College, Andhra Pradesh, India*
[2] *Department of Information Technology, Aurora's Engineering College, Andhra Pradesh, India*
E-mail: sastry_38@yahoo.com, koluguri.87@gmail.com

(Received on: 27-10-11; Accepted on: 12-11-11)

ABSTRACT

Web Applications play a very important role in every person's life. What everyone ignores is the security of the web Application. Enterprises understand the importance of securing web applications [1] to protect critical corporate and customer data. Everyone should concentrate on how to establish security while establishing a web application, i.e. integrating security in traditional development process. As every Web Application should go through Software Development Life Cycle (SDLC) we need to integrate security mechanisms with applications. In this paper we present the mechanism to integrate security mechanism with SDLC of Web Application.

KEYWORDS: *WEB APPLICATION, SECURITY, TESTING.*

1. INTRODUCTION:

Managing the risks associated with complex web applications is a corporate requirement, and the underlying security of the code running these web applications [1][2] directly impacts the risk profile of all corporate data available to the application. Unfortunately, developing repeatable and efficient web application security practices is not an easy task. Many organizations have attempted to provide security controls by using post-production solutions such as web-application firewalls and intrusion prevention systems.[1][3]

But waiting until the production phase of the lifecycle can be "too little, too late." Design or architectural issues that would have been simple to address earlier in the lifecycle become extremely costly to fix once the application is in production. Web application security vulnerabilities lead to data exposure, violations, and can contribute to overall cost when patches or comprehensive code fixes are required after deployment.[1][15]

To be both effective and efficient, web application security must begin with the requirements definition phase carried through code development and implementation, and all the way through to the final acceptance phases. [2] This approach requires that stake holders work together, collaboratively throughout the process as a team. Use of policy-aware automated tools during phases such as implementation and testing, enables repeatable testing and can lead to faster development cycles over time as the testing procedures become standardized. [4]

Building security in from the very beginning of the process doesn't have to be complicated. When security checks and balances are applied throughout the development lifecycle, faster release cycles and a significant reduction in web application vulnerabilities can be achieved.[5]

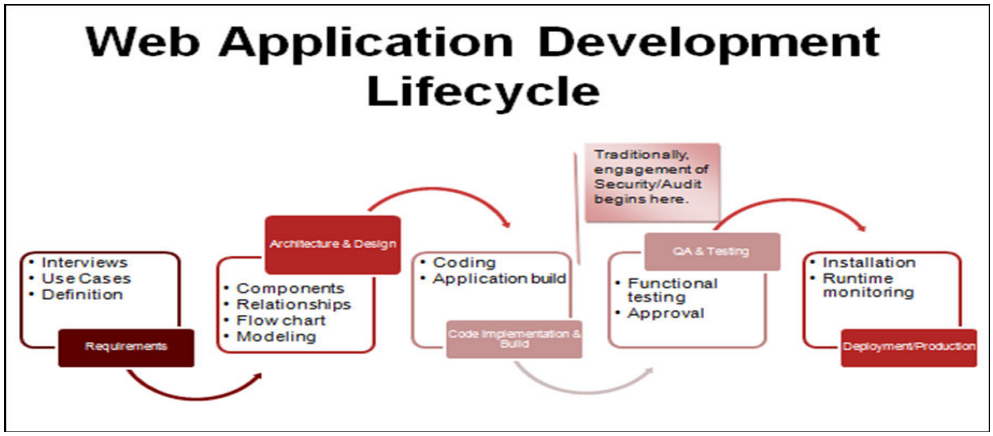

Figure 1: Web Application Development Lifecycle

*Corresponding author: *B. V. S. S. R. S. SASTRY[1]*,*E-mail: sastry_38@yahoo.com*

Though it may sound counterintuitive, adding milestones and security checkpoints to the process really can decrease overall delivery time. The reason is that there are significant costs associated with trying to correct design flaws and coding errors after a web application has been placed into production. [4]

If that sounds extreme, consider this example - a web application that is being built for an insurance firm will manage and store social security numbers (SSN). In the traditional model, security and audit do not see the application until it is being tested prior to deployment. During the security assessment, the security/audit teams discover the SSNs are not encrypted during transmission from the browser client to the web application server; the web application server in turn sends them to a back-end data-base without indicating that the data should be stored in encrypted format.

Because the insurance firm must encrypt SSNs during transit and storage for conformance to disclosure laws and emerging encryption laws related to personal information, the lack of encryption violates corporate policy and governing regulation. The security team asserts that the web application cannot be deployed in its current state, which leads to significant delays and possible business disruption.

The application architect is brought in and determines that portions of the application must be completely re-done to ensure proper SSN protection. The development manager estimates the time for re-write and regression testing will be at least 3 weeks, putting the launch date far behind schedule and significantly increasing the overall cost of the project.

If the security team had been engaged collaboratively from the very first requirements definition meeting, this last minute delay could have been avoided. If the requirements and architecture teams had been able to reference established, reusable corporate policy information about the confidentiality of data in transit, the requirement to protect SSN data would have been known to the architecture and design teams from the get go; it would have been included from the earliest phases as a requirement, built into the web application from the very beginning, and accounted for in the design. If the company had been using policy-aware automated tools within the IDE that checked for SSN encryption, failure to incorporate the encryption control would have been caught during implementation.

When building security in, it's important to keep in mind that the goal in most business scenarios is not to create bullet-proof web applications or even to eliminate every possible exposure. Instead, it is about matching the required properties to the approved risk profile for the web application. The goal, throughout the entire lifecycle should be to achieve "software assurance," that is appropriate to a specific web application's function and sensitivity level, with "justifiable confidence that software will consistently exhibit its required properties. . . . even when the software comes under attack."

2. BENEFITS OF INTEGRATED APPROACH:

Efficiencies in the web application lifecycle result when stakeholders from different groups work together as a collaborative team.[5] While security professionals often lament that business executives don't fully understand software risk, it's just as important that security professionals familiarize them-selves with business risk. Creating web applications with the proper level of software assurance re-quires risk management trade-offs between business needs, usability, and security. To strike the correct balance, requirements input from all stakeholders are required.[6][4]

Starting at the very beginning of the lifecycle, requirements definition and application design take into account security requirements as well as functional and business requirements. This information is communicated to the architects and software developers even before a single line of code is written. [6]This approach will prevent most, if not all, security related design and architecture flaws.

Security-aware software design however will not eliminate all vulnerabilities associated with a web application. Developers themselves must receive training on secure coding techniques to ensure they do not introduce vulnerabilities during the authoring of the application. Training should cover basic security, such as input validation, as well as language specific instruction. Giving developers insight on security practices for the language and run-time environment they are writing applications for supports better coding practices and results in fewer errors in the final web application. Another efficiency benefit of building security in is the ability to build misuse cases during the requirements and design phases. This saves time during testing and acceptance and helps to eliminate bottlenecks.[7]

An integrated, composite analysis approach to testing can increase efficiency even more. Integrated development environment (IDE) specific plug-ins can alert coders as they write if errors are introduced. Static analysis, often called "white box" testing, can be used by developers and auditors on modules before they are assembled into the final product build.[8] Static analysis provides an insider's view of the application at the code level. Static testing is effective at uncovering semantic errors and code level flaws, but not as adept at determining if a flaw will result in an exploitable vulnerability.

Dynamic analysis and manual penetration testing are effective for validating whether or not applications are vulnerable to exploit in production. Often referred to as "black box" testing, dynamic and penetration assessments show the

outsider's view of the application and provide insight into whether or not an attacker would be able to exploit the application in production.[8] However dynamic testing techniques cannot be employed until later in the life-cycle, after

the post-build phase. Another limitation to dynamic testing is that it can be difficult to pinpoint the code level source of the weakness within the code that caused the vulnerability.

This is why it is most effective to use a blend of both static and dynamic testing in a "grey box" or composite approach. By combining results from both the code level insider view and the dynamic outsider view, the strengths of both techniques can be leveraged. Using static and dynamic assessment tools together enables managers and developers to prioritize which applications, modules, and vulnerabilities are the most impactful and need to be addressed first. Another benefit of the composite analysis approach is that vulnerabilities validated by dynamic testing can be sourced back to a specific line or section of code using the static tool. This engenders collaborative communication between the test and development teams and makes it much easier for security and test professionals to provide specific, actionable corrective guidance to developers.[9]

3. IMPLEMENTING SECURITY IN SOFTWARE DEVELOPMENT LIFE CYCLE:

Building security in requires a people, process and technology approach. 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.[11]

The process should include a formal software development lifecycle and published policies. Also important are establishing roles for stakeholders and assigning accountability for review and governance. Security and business should be represented during every phase of the lifecycle so risk management can be addressed at each step.

One constant that is beneficial throughout the entire lifecycle is education. Education has been dis-cussed as important for developers, but it will also benefit every stakeholder involved in web application development. Because security awareness needs to be both top down and bottom up, don't under-estimate the need to educate executives on how web application vulnerabilities can impact the business [12]. Telling an executive that a web application is vulnerable to cross-site request forgery may be met with a blank stare, but showing an executive how software errors can lead to exposure of customer data will help them appreciate the very real consequences of insecure web applications. Provide specific examples and metrics that illustrate potential time and cost savings. For example, demonstrate how investing in developer training and IDE static analysis plug-ins can stop the root cause of data exposure in production before a developer checks in their code for the evening.

Auditors and assessors can benefit from learning about common coding errors, consequence evaluation and dependencies or exposures associated with the web application's eco-system including back-end legacy systems, existing security controls, and any services or applications that are part of the web application's production environment.[13] Testers and QA professionals can be educated about miss-use cases and how they differ from standard-use cases that they might already be familiar with; as well as how to interpret security test findings and prioritize them as needed within the bug fix list.[14]

Looking at specific steps in the lifecycle there are opportunities in each one to increase efficiency while weaving security and risk management throughout.

Let us see in detail about each phase.

3.1. Requirements:

Web application designers are familiar with defining functional and business requirements, but they may not understand how to define security requirements. This is the first opportunity for the teams to work collaboratively to determine what security controls will be essential to the final web application.

Steps for integrating security into the requirements phase:

- Discuss and define security requirements based on corporate policy, compliance and regulatory mandates (for example, two-factor authentication requirement for FFIEC authentication guidance compliance)
- 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

Benefits:

- Security or compliance exposures are eliminated or reduced upfront
- Decreased time-to-deployment
- Significant reduction in acceptance bottlenecks

3.2 Architecture and Design:

As the architecture and design of the web application are defined, security considerations can be as-sessed. It is in this phase that expensive, hard to correct security problems can be fixed at the time they are easiest to address. To prevent costly errors, assess the proposed architecture from both a performance and a security perspective. Detailed design specifications are created, that show develop-ers exactly which security controls must be included and how the components will interact with the overall web application ecosystem.

Steps for integrating security into the architecture and design phases:
Perform risk assessment in context of the application's proposed architecture and deployment envi-ronment, determine if the design will introduce risk

- Assess security implications of interaction with legacy systems and security implications of data "flow" between components, tiers, or systems
- 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
- Communicate the final design to security and audit for finalization of the security test plans and misuse cases

Benefits:

- Fine tuning of risk assessment analysis process and re-usable risk assessment models
- Risks introduced by the architectural environment or deployment context are identified early
- Re-usable misuse cases save time during testing phase
- Reduction in design specific exposures
- If necessary, architectural constraints that introduce risk can be changed and risk mitigation strategies can be defined with compensating controls if the risk cannot be entirely eliminated

3.3 Code Implementation and Build:

When developers begin writing code, they should have a risk assessed design and clear guidance on security controls that must be written into the application or used via an approved service. 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.

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

- Install automated static source code checking tools that are integrated with developer IDEs
- Optionally, developers perform automated code reviews with stand alone coding tools before check-in
- Security and audit teams spot check code modules for compliance conformance and security risk using automated or manual code reviews prior to build
- Implement automated static code scanning during the build process to check for security exposures and policy compliance
- Use tools to track developer coding errors and provide explanatory feedback on security risks introduced and why

Benefits:

- Cleaner/less vulnerable code is delivered to QA
- Developers improve secure coding ability over time
- Re-usable policies increase accuracy of risk analysis
- Fewer coding errors/vulnerabilities discovered during testing resulting in faster deployment cycles

3.4 Testing:

Security specific tools for testing applications range from one-off standalone solutions and services that assess the completed application to fully integrated suites that can provide testing and support from education to implementation and throughout the testing phasing. Integrated solutions provide multi-phase support for companies that are maturing 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.

Key indicators to look for in a solution suite:

- Vendors that understand the entire SDLC and not just one or two phases
- Solutions that can be applied at multiple points throughout the process (education, implementation, testing, deployment)
- Easy to understand, interpret, and use results
- Integration with existing SDLC tools such as build and QA solutions
- Comprehensive reporting for compliance
- Continuous improvement support – identification of developers or application types that require additional training or security controls
- Educational components – online support or training modules

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

- Concentrate on finding the problems that matter most – some resources:
- ✓ SANS Top 25 (http://www.sans.org/top25errors/)
- ✓ CWE (http://www.mitre.org/news/digest/defense_intelligence/02_09/errors.html)
- ✓ OWASP Top Ten Project (http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)
- Validate test findings in a production architecture that includes existing compensating controls such as firewalls and IPS
- 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

Benefits:

- Better communication between application stakeholders
- Fewer false positives
- Faster fix (and release) cycles

3.5 Deployment/Production:

Web application security doesn't end when an application is deployed. 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 applications provides assurance that the web application is performing as expected and not exposing information or introducing risk. Monitoring can be done by in-house staff or outsourced to an external provider that can monitor the application(s) on a 24/7 basis.

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

- Monitor misuse to affirm vulnerabilities deemed "not exploitable" in testing are not exploitable in production
- Monitor data leakage to look for places where is used, sent or stored inappropriately
- Compare pre-deployment residual risk assessments with in-production exposure areas and provide feedback to testing team
- Implement web application firewall, IPS, or other compensating control to mitigate exposures before code fix or in response to new regulations or compliance mandates

Benefits:

- Improve knowledge-base of successful exploits to improve scan efficacy during static and dynamic tests
- Find and stop unexpected misuses of the application, even during incremental refreshes
- Better integration between dynamic testing and production application controls such as web-application firewalls or IPS
- Meet emerging compliance requirements before code re-write
- Continuous improvement – using the feedback loop

4. CONCLUSIONS:

Web application security is achievable in a time-sensitive manner when stakeholders work together collaboratively. Securing web applications doesn't have to mean extending lifecycles or major disruption to the development process. With education of all stakeholders and a clear, repeatable process, organizations can incorporate security and risk throughout the lifecycle in an efficient, cooperative way.

Weaving security into the web application delivery lifecycle does require a synergistic approach that incorporates people, process, and technology. Though web application security tools and suites can contribute to process improvement, they are not a panacea. For maximum benefit, look for web application security tool vendors that understand the whole development lifecycle and have tools that provide support at multiple stages of the process.

ACKNOWLEDGEMENTS:

We, the authors would like to thank everyone, who supported in the preparation of the paper Specially Dr. S.M Afroz, HOD CSE Department of Nizam Institute of Engineering & Technology who has Guided us for preparing this paper.

REFERENCES:

[1] Developing Secure Web Applications by Izhar Bar Gad.
[2] Developing a Secure Web Application from a coding perspective by Jamie Spicatti.
[3] Guide to Secure Web Services by Anoop Singhal.
[4] Securing Web Applications through a Secure Reverse Proxy by Anh-Duy Ngyuen.
[5] Developing Secure Web Applications by Scott.D and Sharp R, IEEE Internet Computing 2002, pg.38-45.
[6] Security in Coding phase of SDLC by Kumar.R, IEEE 2007, pg.118-120.
[7] Techniques for Quantitative Analysis of Software Quality throughout the SDLC by Talib, IEEE 2010, pg. 321-328.
[8] Software Process Engineering by Emami, IEEE2010, pg:1-5.
[9] Best Practices for Software Security by Yasar, IEEE 2008, pg.169-173.
[10] Role of Software Metrics in Software Engineering and Requirements Analysis by Durrani, ICICT 2005,pg.71.
[11] Securing Multi – tiered Web Applications by Mathew, IEEE 2010, pg.505-509.
[12] Characterizing Secure Dynamic Web Applications Scalability by Guitart, IEEE 2005, pg.108.
[13] Secure Web Based Applications with XML and RBAC by Yang, c, IEEE 2003, pg.276-281.
[14] Securing Timeout Instructions in Web Applications by Russo.A, IEEE 2009, pg.92-106.
[15] A study of Security and Performance Issues in Designing Web Based Applications by Shin-Jer Yang, IEEE 2007, pg.81-88.

Authors:

B. V. S. S. R. S. Sastry has been graduated with B. Tech in 2009 from Aurora's Engineering College, Bhongir, Andhra Pradesh, India. He is currently pursuing M. Tech from Aurora's Engineering College, Bhongir, Andhra Pradesh, India. Contact him at **sastry_38@yahoo.com.**

K. Akshitha has been graduated with B. Tech in 2009 from Royal Institute of Technology and Science, Chevella, R. R. Dist, Andhra Pradesh, India. She is currently pursuing M. Tech from Aurora's Engineering College, Bhongir, Andhra Pradesh, India. Contact her at **Koluguri.87@gmail.com.**

****

