SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 1, ID. SAUCIS-4-2018, APRIL 2018

Developer-oriented Web Security by Integrating Secure SDLC into IDEs

Emin İslam TATLI

Emin İslam Tatlı; Corresponding Author; emin.tatli@stm.com.tr; 0 (312) 266 3550
STM, Director of Cyber Security and Big Data

Received 17 March 2018; Accepted 30 March 2018; Published online 2 April 2018

Abstract

Enterprises and organizations have difficulties to protect their web-based services against cyber-attacks. Due to increasing number of cyber-attacks, critical data including customer data, patient data etc. are leaked and critical services like online banking become unavailable for long period of time. The studies of Gartner, OWASP, SANS and similar organizations have shown that today's cyber-attacks target mostly application layer. This means that application developers design and implement insecure web applications and black-hat hackers exploit these security weaknesses to get unauthorized accesses to critical databases. Insecure development of web developers is still a big challenge to solve. The top one risk "SQL Injection" from OWASP Top 10 list can be given as a concrete example. This vulnerability was discovered 20 years ago, but web developers are still mostly unaware of its prevention methods. The weak communication between web developers and security experts is one of the main reasons of insecurely developed applications. Even though security experts have the knowledge of all preventions methods for all types of security vulnerabilities, they are insufficient to transfer this knowledge to web developers. Secure software development lifecycles methodologies like Microsoft SDL, OpenSAMM, BSIMM have been also proposed in order to integrate required security activities into all phases of software development. But the security activities required by these methodologies are not integrated within development environments and therefore secure coding awareness of developers cannot be efficiently achieved. In this paper, we suggest new methods and discuss open academic research issues for integration of secure SDLC activities including secure coding practices and secure architecture patterns into development IDEs (Integrated Development Environments). Providing this, web developers can access to secure coding procedures and best-practices directly within their IDEs, increase their security awareness and develop more secure applications. As a result, the numbers of security vulnerabilities would drastically decrease and critical data leakages can be prevented.

Keywords: *secure coding; secure software development; secure SDLC*

1. Introduction

Information security is one the most critical and difficult challenges in the Internet era. Organizations and enterprises providing services through web applications have difficulty to secure their IT systems and sensitive data against cyber-attacks. Every day several enterprises become victim of cyber-attacks (Hackmageddon 2018) by which sensitive data (e.g. customer data of ecommerce companies, patient data at hospitals, employee data, etc.) are seized or critical services (e.g. online banking, online shopping etc.) become unavailable for very long time. All these attacks and data breaches result into financial lost and damage to customer trust. Gartner estimates that 70% of cyber-attacks target the application layer. That means developers design and implement insecure web applications, black-hat hackers find vulnerabilities within such applications and exploit them to get unauthorized access to databases and servers. The reason why developers produce insecure web applications is a big dilemma. For example, SQL injection, the most critical web security vulnerability according to OWASP (Open Web Application Security Project) Top 10 project (OWASP-Top10 2017), was discovered more than 15 years ago. Even though SQL injection and its prevention methods are known since then, today many developers are unaware of them and their applications are vulnerable to SQL injection attacks.

The dilemma of insecure applications is that there is a huge gap between security experts and web developers. Security experts know every type of vulnerabilities in detail as well as tools, libraries, best-practice coding methods, secure configurations etc. as the solutions, but they are very unsuccessful to transmit their knowledge to be utilized by web application developers.

In this paper, a developer-oriented web security model that integrates secure SDLC activities into IDEs is proposed. Exploiting this model, developers can be involved with the lifecycle of secure software development easily, apply security policies and requirements during coding more accurately, exploit secure architecture design best-practices and technology-dependent methods for secure coding more efficiently, understand the details of security vulnerabilities identified by penetration testers and have a better knowledge and capability to fix the identified security vulnerabilities. The proposed model combines practical development experiences with academic research aspects of security and software engineering and offers solutions for real-life development projects to make the Internet a more secure place.

The paper is organized as follows: Section 2 explains the problems and challenges of secure software development from the perspectives of web developers. Section 3 lists and explains the requirements for secure SDLC integration into IDEs. Open issues for academic research regarding secure SDLC integration into IDEs are explained in Section 4. This section highlights possible new topics for security researchers. Section 5 discusses the related works. Section 6 concludes the paper.

2. The Problems of Secure Software Development

Web applications have become very popular in the Internet era since they can be developed very rapidly and are accessible through all browsers of any operating systems. Security, on the other hand, is a big headache for web applications. OWASP Top 10 project lists the most critical security risks relating to web applications since 2004. The number one risk "Injection" allows attackers to send untrusted data to a targeted interpreter (e.g. SQL, LDAP, XPath interpreters) and manipulate the interpreter's code execution. For example, attackers performing SQL injection attacks can get unauthorized access to critical databases and even manipulate table entries.

Since the beginning of 2000s, security researchers have analyzed web application flaws and proposed solutions. But today there are still many web developers who have not heard about SQL injection and have no idea how to prevent it. Being unaware of the security risks, developers design and implement insecure applications which expose sensitive data to hackers and threaten the businesses of their organizations and enterprises. The main reason of this dilemma is that security experts are insufficient to transmit their experiences to developers. This dilemma can be explained further with the following key issues:

- Most organizations do not have sufficient number of security professionals that are responsible for application security tasks. Besides, the existing secure SDLC methodologies (e.g. OpenSAMM, BSIMM etc.) are too complex and heavy for such organizations to apply. A lightweight SDLC should be designed and developed.
- Organizations and enterprises having a security management system in place require their developers to perform certain activities based on their secure software development lifecycle (SSDLC). But developers are mostly not trained about the SSDLC process of their organizations, do not know which security activities must be performed at each development phase and how to perform. Therefore, they ignore the SSDLC process.
- Organizations and enterprises define their security rules and publish them within a security policy document. But these security policies remain untouched within the policy document and are not taken into consideration by web developers during coding. They are not supported by security experts to understand the security policies and apply them during development. They ignore security policies too.
- It is expected that threat analysis is performed especially for confidential, integrity-critical or/and availability-critical classified systems. Threat analysis shows all possible threats from the

perspectives of different attackers and entry points. Considering all threats, the required security controls are identified, implemented and enforced. But web developers are not familiar with threat analysis and many critical threats are ignored too.

- Web framework developers are not sufficiently qualified to integrate security controls within their frameworks. Providing this security integration, many critical flaws could be automatically prevented without any interaction of application developers utilizing these frameworks.
- Developers are occupied with many text-based documents about secure coding, security policies, secure software development lifecycles, penetration testing reports etc. Instead, checklist-based tools can help them to understand the security requirements better and perform their tasks more easily and efficiently.
- Mostly automatic web application scanner (WAS) tools are used for security testing. Manual testing and developer trainings are mostly ignored. But it is overlooked that WAS tools are not able to detect business logic errors (BLV 2015).
- Penetration testers perform black-box and white-box security checks and generate pdf reports which explain their findings and the countermeasures to fix them. Web developers getting a pdf report do not understand many countermeasure details and are not able to fix the vulnerabilities.

3. Requirements for Developer-oriented Web Security

Since security must be considered "as a process rather than a product" (Schneier 2000) it is vital to integrate security activities into each phase of software development. This means web developers should consider and perform certain security activities at each phase of software development. Considering this fact, several methodologies (e.g. (OpenSAMMv1.5 2017; BSIMMv8 2017; Microsoft SDL 2010) etc.) have been proposed in the past. These methodologies are called as Secure Software Development Lifecycles (SSDLC). Figure 1 illustrates the main architecture of OpenSAMM which requires performing security activities of twelve security practices that are categorized under four business functions. Each security practice consists of three maturity levels and each maturity level consists of several security activities.

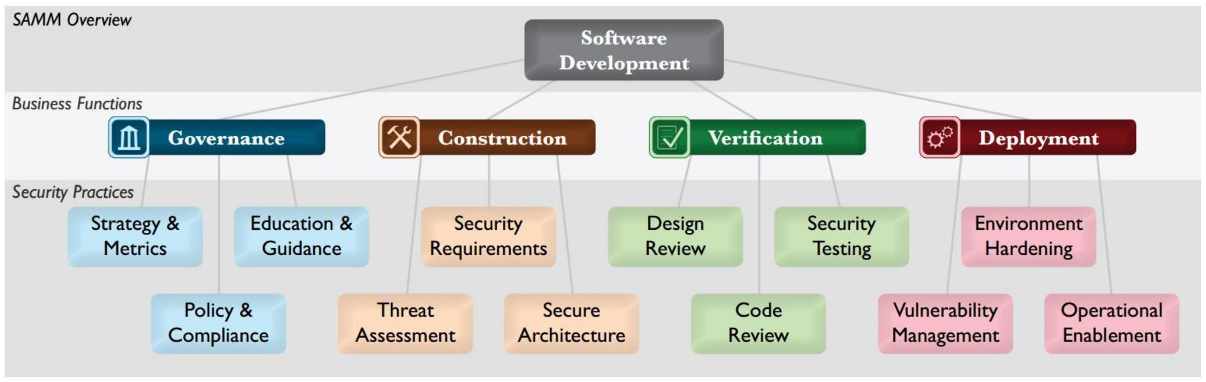

Figure 1 Business Functions and Security Practices of OpenSAMM

Examining the twelve security practices of OpenSAMM, it is realized that security activities related to security trainings, security policies, SSDLC methodology, threat analysis, security requirements, security architecture, design review, black-box and white-box security testing, environment hardening, vulnerability management and operational documentation are required to be performed during software development. It is certain that web developers cannot understand and follow such a complicated process easily. In order to help web developers by secure development, their IDE platforms should be equipped with supporting tools, supporting processes and training materials.

In the following, the requirements of secure SDLC integration into IDEs are discussed in detail:

i. Development of a lightweight secure SDLC

Due to lack of security experts at most organizations, existing secure SDLC methodologies (e.g. OpenSAMM, BSIMM etc.) are too complex and heavy for these organizations to apply. A new lightweight methodology that is developed from web developers' perspective is needed. This new methodology should be optimized and contain only the most important and relevant security activities with new optimized maturity levels.

ii. Integration of secure SDLC process into development environments

It is required that an organization's secure SDLC process is integrated into development process and environment. This integration is very crucial for all other security practices of SDLC. Providing this, web developers and development project members can directly access their secure SDLC process and get detailed information about each security activity they need to perform at each development phase. They can access supporting tips and tools for each security activity including secure design requirements and coding best-practice examples. This integration should be project management oriented. That means a project manager can assign roles and tasks to project members and see all performed secure SDLC activities. Project members can set status of each assigned security activity and follow status of their overall security tasks within the project in a more structured way.

iii. Enhanced management of requirements and threats

There exist several requirements (e.g. legal, security policy and best-practice aspects) that need to be considered during development. But web developers have difficulty in extracting security requirements from text documentation. Therefore, legal, security policy and best-practice design and implementation requirements as check-lists should be integrated into development environments. This aspect is relevant for Security Requirements, Security Architecture, Policy&Compliance, Threat Assessment and Design Review security practices.

Providing this, web developers can easily access the security requirements for each development phase (e.g. design, implementation, test) and understand their tasks via additional hints. Special check-lists should be provided for different project roles like system administrator, database administrator, penetration tester, project manager, etc. The requirement check-lists should be created according to information classification (i.e. confidentiality, integrity and availability parameters).

iv. Technology-specific support during coding

Web developers should access directly technology or framework specific best-practice examples during coding. This aspect is relevant for Secure Architecture and Design Review security practices. Providing this, web developers can be provided with any necessary information on the fly during coding. As an example, a web developer implementing in Java can access PreparedStatement Java code example directly within the IDE to learn how to prevent SQL injection.

v. Manageable vulnerability management process

Rather than focusing on pdf reports, an enhanced process should be provided to web developers for dealing with and fixing vulnerabilities in a more efficient way. This aspect is relevant for Security Testing and Vulnerability Management security practices. Providing this, vulnerabilities identified in penetration tests can be presented to developers as online accessible check-lists rather than as pdf reports. Web developers can get more structured information about each vulnerability and understand fixing instructions easily. Organizations having thousands of web applications can analyze their vulnerabilities in detail and manage fixing process more efficiently.

vi. Exploiting framework security features

Web developers and system administrators are not familiar with which security features and configurations web frameworks support. Easy access to security features and configurations of web development frameworks (e.g. Spring, Struts, JSF, .NET etc.) should be possible to activate and use them more efficiently. This aspect is relevant for Secure Architecture and Environment Hardening security practices. Providing this, framework security features can be integrated into development environments and become accessible by web developers easily and more efficiently.

vii. Enhanced security code review

Manual security code review is required especially for security-sensitive classified applications. Mostly search functions of IDEs are used during code review. But this is not a safe method since a reviewer alone is not eligible to memorize all critical classes and methods. Therefore, an enhanced method with a tool which presents code reviewers the critical classes and methods of different technologies to check should exist. This aspect is relevant for Code Review security practice. Providing this, using this new tool, code reviewers can know exactly which classes and methods are critical to check.

viii. Detecting business logic flaws

Web Application Vulnerability Scanner tools are not able to detect business logic flaws. Academic research needs to be carried out to close this gap. Business logic flaws can be categorized and scanning methods can be developed to search weaknesses within certain categories of business logics. For example, negative price by e-commerce applications is a typical type of business logic errors. Price parameters can be checked *automatically* for negativity. This aspect is relevant for Security Testing security practice.

4. Open Issues for Academic Research

Considering the requirements explained in the previous section in detail, open research topics that need to be analzyed and matured in the academia further are discussed in this section. Despite its importance and impact, secure SDLC related academic research has been so far carried out very limited in the academia. Table *1* lists possible open research topics of secure SDLC for academic research. It shows a roadmap for security researchers that are interested in secure development.

Table 1. Open Research Topics for Developer-oriented Web Security Framework

Task	Open Research Topics
Development of a lightweight secure SDLC	– Existing secure SDLC methodologies need to be studied, analyzed and compared in detail. – The most required security practices and activities from the secure SDLC methodologies need to be identified.

Task	**Open Research Topics**
	– Unessential security practices and activities need to be eliminated. – The maturity levels (between 1-3) need to be re-organized accordingly.
Integration of secure SDLC process into development environments	– The main architecture for integration needs to be designed by studying its features and integration aspects. – The requirements and best methods need to be investigated to integrate each security practice of secure SDLC into IDEs. – IDEs (e.g. Eclipse, NetBeans, Visual Studio etc.) need to be analyzed in detail for this integration. Especially, usability issues need to be considered for the integration since there is a trade-off between usability and security. – Moreover, each organization has its own customized secure SDLC. It should be examined how to achieve a flexible architecture in order to integrate and support any type of secure SDLC process.
Enhanced management of requirements	– It should be investigated how and which existing security requirements are relevant for secure coding and how they can be integrated into IDEs. – Security policies include legal and organizational requirements in addition to technical requirements. It should be analyzed how legal and organizational requirements can be integrated into IDEs. – It should be analyzed how information classification process and maturity levels are covered within integration process. – It should be investigated how requirement check-lists for different roles and for different development phases can be created and integrated.
Improvement and integration of threat assessment	– Existing threat assessment methods (e.g. Attack-tree (Saini et al. 2008), Attack-Protection tree etc.) and tools need to be analyzed from the perspectives of web developers. An improved methodology needs to be developed and integrated into IDEs.
Technology-specific secure coding support	– It should be analyzed which technology and framework specific secure coding examples and patterns can be integrated into IDEs while considering usability trade-off.
Enhanced vulnerability management process	– Vulnerability management including task assignment, fixing and reporting processes should be analyzed. – An enhanced vulnerability management system should be developed and integrated into IDEs.
Exploiting framework security features	– Existing web frameworks (e.g. Spring, Struts, Django, Zend, AngularJS, EmberJS etc.) should be studied to identify and compare supported security features and configurations.

Task	Open Research Topics
	– It should be examined how supported framework security features can be integrated into IDEs. – Moreover, it should be analyzed which new types of security features can be integrated into each studied web framework to prevent cyber-attacks automatically.
Enhanced security code review	– Code review tools need to be equipped with the list of critical classes and methods from the security perspective. – Critical classes and methods for different technologies (e.g. Java, dotNET, PHP etc.) need to be identified. – It should be then investigated how to integrate security code review functional into IDEs. – Tatlı and Urgun developed a manual security code review tool namely *ccrawl (*Tatlı and Urgun 2013*)* as an initial version of such a required tool.
Detecting business logic flaws	– Business logic flaws should be studied in order to identify different categories of business logic flaw types. – Methods for scanning weaknesses within the identified categories should be developed. – Tools implementing these detection methods should be developed and integrated into IDEs.

5. Related Work

There are several academic works that aim to support secure SDLC processes at the beginning of requirements or design phases. Talukder et al. developed a self-alone tool that can be exploited for different tasks within secure SDLC (Talukder et al. 2009). Using the tool, asset lists with information classification can be created, abuse-case and attack tree diagrams can be created and several testing tools can be accessed. This tool was not developed from the perspective of developers. It only provides GUIs for existing threat analysis methods and gives references for several existing test tools.

Zenah and Aziz developed a web-based training tool for developers which contain several exercises based on OWASP Top 10 and SANS 25 (SANS 2011) lists (Zenah and Aziz 2011). Thuraisingham and Hamlen discuss challenges and future directions of secure software development regarding formal methods and secure service modelling (Thuraisingham and Hamlen 2010). Bilge and Dumitras performed a benchmark analysis and showed the time difference between discovery time and fixing time of vulnerabilities (Bilge and Dumitras 2012).

SecureUML (Lodderstedt et al. 2002) and UMLSec (Jürjens 2004) introduce model-driven security engineering methods and enable integration of security requirements (e.g. authentication, authorization, etc.) within different UML diagrams. Mouratidis and Giorgini propose an approach called Security Attack Testing (SAT) for testing the security of a software system during design (Mouratidis and Giorgini 2007). SAT helps developers to understand the goals of possible attackers based on possible attack scenarios.

Adebiyi et al. propose a neural-network based tool to evaluate the security of software design at the design phase of SDLC (Adebiyi et al. 2012). Othmane et al. propose a new method that integrates security engineering activities into the agile software development process (Othmane et al. 2014).

Considering all related works, the proposed project offers practical as well as theoretical contributions. The related works propose mostly solutions for a limited part of a secure SDLC process. This makes them inapplicable for real-life development projects since they overlook the big picture regarding all security activities of secure SDLC process.

6. Conclusion

Most web application developers are not experts for secure development and coding. As a result, most web applications today contain critical security vulnerabilities. Security experts know already the solutions to prevent these security vulnerabilities, but the knowledge transfer between web developers and security experts is today very problematic. In order to close this gap, we suggest that secure SDLC methodologies are integrated into developers' IDE platforms. Secure SDLC requires that several different security activities should be performed in all phases of software development (i.e. requirement, plan, design, implementation, testing, and operation). In this paper, we have proposed a model to enable this integration. The requirements as well as the open issues for academic researches are explained and discussed for the proposed model in the paper in detail.

References

Adebiyi, A., Arreymbi, J., Imafidon, C. 2012. "Applicability of Neural Networks to Software Security", 14th International Conference on Computer Modelling and Simulation, 19-24.

Bilge, L., Dumitras, T. 2012. "Before We Knew It: an Empirical Study of Zero-Day Attacks in the Real World", Proceedings of the 2012 ACM Conference on Computer and Communications Security -- CCS'12, 833–844.

BLV, "Business logic vulnerability". https://www.owasp.org/index.php/Business_logic_vulnerability, August 2015.

BSIMM (The Building Security In Maturity Model) v8, https://www.bsimm.com/, September 2017.

Hackmageddon – Cyber Attacks Timelines and Statistics, http://hackmageddon.com, 2018.

Jürjens, J. 2004. Secure Systems Development with UML. Springer-Verlag.

Lodderstedt, T., Basin, D.A., Doser, J. 2002. "SecureUML: A UML-Based Modeling Language for Model Driven Security," In Proceedings of the 5th International Conference on the Unified Modelling Language, 426-441.

Microsoft SDL, https://www.microsoft.com/en-us/sdl, 2010.

Mouratidis, H., Giorgini, P. 2007. "Security attack testing (SAT)- testing the security of information systems at design time", Information Systems, Vol. 32, 1166-1183.

OpenSAMM (Software Assurance Maturity Model) v1.5, http://www.opensamm.org/, 13 April 2017.

Othmane, L.B., Angin, P., Weffers, H., Bhargava, B. 2014. "Extending the Agile Development Approach to Develop Acceptably Secure Software", Dependable and Secure Computing, IEEE Transactions on , vol.PP, no.99, 497 - 509.

OWASP-Top10 Project, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, 2017.

Saini, V., Duan, Q., and Paruchuri, V. 2008. "Threat modeling using attack trees". Journal of Computing Sciences in Colleges (APRIL), 124–131.

SANS, "25 Most Dangerous Software Errors v3". https://www.sans.org/top25-software-errors/, June 2011.

Schneier Bruce, "The Process of Security", http://www.schneier.com/essays/archives/2000/04/the_process_of_secur.html, April 2000.

Talukder, AK., Maurya, V.K., Santhosh, B.G., Jangam, E., Muni, S.V., Jevitha, K. P., Saurabh, S., Pais, AR. 2009. "Security-aware Software Development Life Cycle (SaSDLC) - Processes and tools," Wireless and Optical Communications Networks.

Tatli Emin İslam and Urgun Bedirhan, Ccrawl-a thick client helping security static code review processes, https://github.com/agguvenligi/ccrawl, October 2013.

Thuraisingham, B., Hamlen, K.W. 2010. "Challenges and Future Directions of Software Technology: Secure Software Development", IEEE Computer Software and Applications Conference

(COMPSAC), 17-20.

Zenah, N.H.Z., Aziz, N.A 2011. "Secure coding in software development," Software Engineering (MySEC) 5th Malaysian Conference, 458-464.