Asian Journal of Information Technology 16 (1): 7-13, 2017
ISSN: 1682-3915

Secure Web Application Development Prototype Using Enterprise Security Application Programming Interface (ESAPI)

[1]Abdul Barakath Mohamed Rasheed, [2]Bharanidharan Shanmugam, [1]Ganthan Narayana Samy, [1]Nurazean Maarop, [1]Pritheega Magalingam, [2]Khar Cheng Yeo and [2]Sami Azam
[1]Advanced Informatics School (AIS), Universiti Teknologi Malaysia, Johor Bahru, Malaysia
[2]Casuarina Campus, Charles Darwin University, Casuarina, Australia

Abstract: The web application has been playing a key role in the development of modern society. Unlike traditional applications, modern web applications are generally more exposed to untrusted users, data and transmission medium. According to a cenzic 2014 report 96% of all applications tested in 2013 have one or more serious security vulnerability. The root causes behind these vulnerabilities are lack of application security awareness, design flaws and secure coding. Furthermore, developers frequently see functionality as more important than security. Therefore, this study proposed a simple implementation of the single security Application Programming Interface (API) that could minimize web application security flaws and prevent from critical malicious attacks. A prototype application is developed with open web Application Security Project (OWASP) enterprise security application API based on Rapid Application Development (RAD) methodology. Thus, this study been carried out with an aim to fill the gap between web application development and application security domain.

Key words: Web application security risks, Rapid Application Development (RAD), Application Programming Interface (API), Enterprise Security Application Programming Interface (ESAPI), Open Web Application Security Project (OWASP)

INTRODUCTION

The web application has become more and more critical in every domain of human society such as communications, e-Commerce, education, health, military, entertainment and others. Not only government sectors and major corporation are using these applications but also across all organizations and even by individual users. As most of them rely on websites to interact with employee, deliver content to their customers, provide information to the public and sell products certain technologies are often deployed to handle the different tasks in the web application. Therefore, web application even becomes a strategy's core competency for any organization.

Web application vulnerabilities can be costly for the organization which may include direct financial losses and tarnished image and brand. Organization security strategy often includes development process model and choosing tools that reduce or mitigate the various vulnerabilities present in the web applications.

However, software developers frequently see functionality as more important than quality or security. This is natural since the functionality is what represents the need for a given product. Without it there is even no need for security because there will be nothing to be secured (Kaur *et al*., 2012). The study showed that almost every company is conscious of the risks of insecure software and performs all these activities to some extent. Although their clients rarely or do not ask explicitly for security, software-house developers implement security assurance mechanisms because they are aware that in case something goes wrong it will bring them negative consequences (Epstein, 2009).

In order to obtain a secure web application, the typical development approach needs to be extended. Today, web application developers need to be skilled in diverse areas of discipline. It's essential to build an application which is user friendly, highly performance, accessible and most importantly is secure while executing and communicating via untrusted environment that the developer or organization has no control over it.

One key area of the problem is that application security is a large and complex discipline that most developers will not be able to master in a short time frame, particularly if they are busy getting and understanding

Corresponding Author: Abdul Barakath Mohamed Rasheed, Advanced Informatics School (AIS), Universiti Teknologi Malaysia, Johor Bahru, Malaysia

the requirement, designing the application and even to learn the latest framework or language. The road forward initiative must be to simplify the implementation of security domain in the application development process for developers.

Therefore, the first objective of this research is to identify the usage of Enterprise Security Application Programming Interface (ESAPI) in a software development process that can minimize the software vulnerabilities. The second objective is to develop a prototype by effectively adopting ESAPI accommodating application security domain. The last objective is to evaluate the effectiveness of (ESAPI) in the web application through penetration testing.

Literature review: It is first important to present the common software development models before addressing the process involves in web application security. A software development methodology refers to the framework that is used to plan, structure and control the process involves in developing software. Varieties of models have evolved over the years, each with its own strengths and weaknesses. Some of the most well-known models are the waterfall model, rapid application development and the spiral model (Fonseca *et al*., 2013).

Basically, one software development methodology is not necessarily suitable to be used by all projects. Each of the available frameworks is best suited to specific kinds of projects which based on various organizational, project, team and technical considerations.

The first structured approach in software development was a waterfall model. It gained in popularity and been used widely during the 70's and structured for a certain orders or steps in the process. It is just a time-ordered list of activities to be performed to develop a system. Basically, waterfall model was and remains popular until now a days.

However, it is quite rare to meet software projects that follow the model's steps exactly. In addition, unlike prototyping or other methodologies, it does not allow for feedback loops. This means that for example, once the requirements are written, they should be stable throughout the project. Any reviewing of them is thus an extraneous and unplanned activity (Siddiqui and Hussain, 2006).

The strengths of waterfall model is the capability of minimizes planning overhead since it can be done up front. The structure reduces wasted effort, so it works fine for inexperienced staff or technically weak. However, it is inflexible since only the final phase produces a non-documentation deliverable. Backing up to address mistakes is difficult (Munassar and Govardhan, 2010). The most appropriate situation to implement the waterfall model are for large software projects where the requirements are not well understood or allow for any reasons of changes such due to external changes, expectations, budget or rapidly changing technology.

The strengths of the Rapid Application Development (RAD) model are to produce software more quickly while to a business perspective, RAD approach is likely to produce software at a lower cost. The V-shaped model is simple and easy to use. Each phase has specific defined deliverables. The early development of test plans during the model life cycle lead to a higher chance of success compared to the waterfall model. It works fine for a small project where requirements are easily gathered and understood. However, adjusting scope is difficult and expensive, very rigid and has the only little flexibility using this model. It does not provide a clear path for problems found during testing phases (Munassar and Govardhan, 2010).

The strengths of Extreme Programming (XP) are basically, it is a lightweight method which suits small to medium projects. It produces good team cooperation and unity while emphasizing final iterative product. A test based approach applied for requirements gathering and quality assurance. However, this model is difficult to extend up to a large project which required and stress on documentation. Programming pairs are costly. Test case construction is a difficult and need specialized skill (Munassar and Govardhan, 2010).

Comprehensive, Lightweight Application Security Process (CLASP) consists of components with formalized security best practices that cover the entire Software Development Lifecycle (SDLC) so that security concerns can be adopted from the early stages of the SDLC used by the organization. This set of 24 security correlated activities is easily integrated into the SDLC of the application allows systematically addressing security vulnerabilities.

The Microsoft Secure Software Development Lifecycle (SSDL) is based on a set of activities for each phase and it can be applied incrementally into an existing ongoing development process. The Microsoft SSDL is based on the following guiding principles: security by design, security by default, security in development and communications.

The information security community has also evolved since the early stages of web application development. Organizations such as the Open Web Application Security Project (OWASP) have expanded and have been involved in a large number of projects to promote many different aspects of web application security from risk

Table 1: OWASP top ten most critical web application security flaws 2013

Ranks	Vulnerability	Description
A1	Injection	Injection fiaws such as SQL, OS and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization
A2	Broken authentic and session managent	Application function related to authentication and session management are often not implimented correctly, allowing Attackers to compromise passwords, keys or session tokens or to exploit others implementation flaws to assue other user's identities
A3	Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victm's browsr which can hijack user sessions, Deface web sits or redirect the user to malicious sites
A4	Insecure direct object references	Adirect object reference occurs when a develpoer exposes a reference to an internal implementation object such as a file, director or database key. Without an access control check or other protection, attackers can maniqulate these references to access unauthoriz data
A5	Security misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server and platform. Secure setting should be defined, implemented and maintained as defaults are defaults and ofern inserure. Additonally, sofware should be kept up to data
A6	Sensitive data exposure	Many web application do not properly protect sensitve data such as credit cards, tax IDs and authentication credential. Attackers may steal or modify such weakley protected data to conduct credit card fraud, identity theft or other crimes. Sensitive data deserves extra protection such as encryption ata at rest or in transit as well as special precautions when exchanged with the browser
A7	Missing function level access control	Most web application verify function level access rights before meking that functionality visivle in the UI however, applications used to perform the same access control checks on the servers when each function is acessed. If requests are not verifed. Attackers will be able to forge requests in order to access functionality without proper authorization
A8	Cross-Site Request Foegery (CSRF)	A CSRF attack forces a logged on victim's browesr to send a forged HTTP requst including the victim's session cookie and any other automatically ancluded authentication information to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim
A9	Using compponent with known vulnerabilities	Componets such as libraries frameworks and other software modules, almost alwayes run with full privileges if a vulnerable components is exploited such an attack can facilitate serious data loss or server takeover. Application using components with known vulnerabilities may undermine applicatin defense and enable a rang of possible attacks and impacts
A10	Invalidated redirects and foewards	Web application frequently redirect and forward users to other pages and website and use untruseted data to determine the destination pages. Without proper validation, attackers can redirect vivtims to phishing or malware site or use forwrads to accass unauthorized page

assessment guides to security testing tools (Martin, 2011). OWASP top ten aims to provide a list of the most critical web application security risks and to educate about the consequences of the most important web application security weaknesses to all respective parties among developers, designers, architects, managers and organizations. Table 1 explains the OWASP top ten most critical web application security flaws 2013.

Basically, the open web Application Security Project (OWASP) is a worldwide not-for-profit charitable organization focused on improving the security of software. The OWASP organization maintained a large number of projects in varying levels of support, ranging from documentation (top ten) to teaching tools (webgoat) to proxies (webscarab) to a secure development library (ESAPI) and so on. For example the OWASP Top Ten project is a list of the 10 most critical issues in the web application security. The list is updated every few years, most recently in year of 2013 and is compiled with lots of input from various industries.

The typical development approach needs to be extended to obtain a secure product. The OWASP's Enterprise Security API Project (ESAPI) addresses this problem by providing a set of Application Programming Interface (API) to mitigate all security controls needed to build a secure application including input validation, output encoding, error logging and detection. These API include toolkits for the major programming languages such as Java, PHP and Phyton and can be used by software developers during the SDLC to increase security with minimum intrusion in their development process.

Enterprise Security Application Programming Interface (ESAPI) is a OWASP Enterprise Security API whereby it is a free and open source web application security control library that makes it easier for programmers to write lower-risk applications. Besides serve as a solid foundation for new development. The ESAPI libraries are also designed to make it easier for programmers to retrofit security into existing applications. Some other researchers propose secure development guidelines like the OWASP's guideline on building secure Web applications and web services.

MATERIALS AND METHODS

This research is building based on applied research methodology as presented in Fig. 1. Basically, applied research aims at finding a solution to an immediate

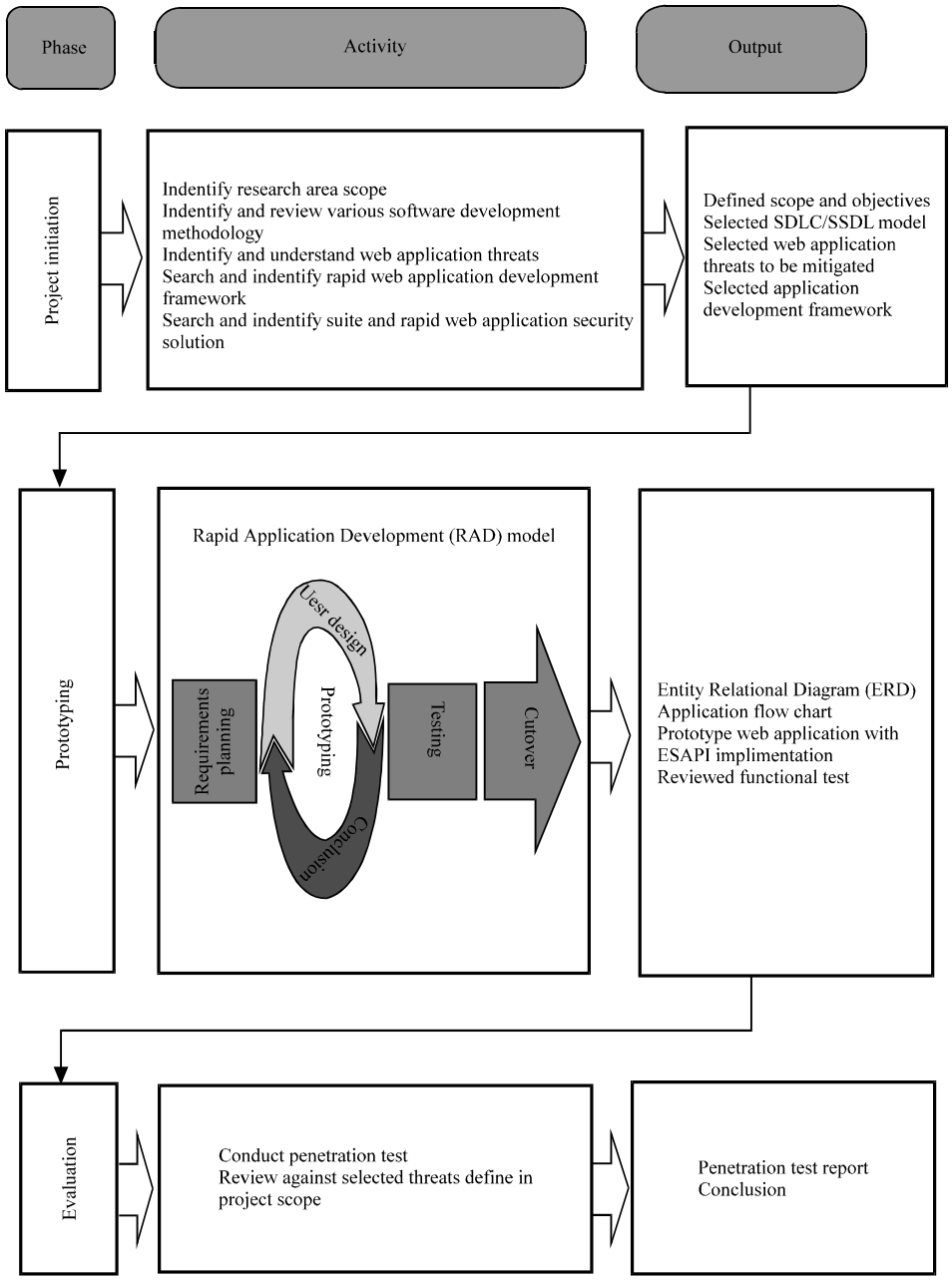

Fig. 1: The research framework

problem facing a society or an business organization (Kothari, 2010). In this research, a Rapid Application Development (RAD) model is selected. Basically, Zed Attack Proxy (ZAP) is an OWASP flagship project is selected to be used for web application penetration test on this stage. Besides ZAP developed as open source software, it is also easy to be used and had a wide verity of penetration test features. A test plan is constructed using ZAP to test the prototype application vulnerabilities.

RESULTS AND DISCUSSION

Implementation and results: The OWASP top ten project is a list of the ten most critical issues in the web application security. OWASP top ten project primary aim is to educate web application developers, designers, architects, managers and organizations about the consequences of the most important web application security weaknesses. It provides basic techniques to protect against these high risks problem areas.

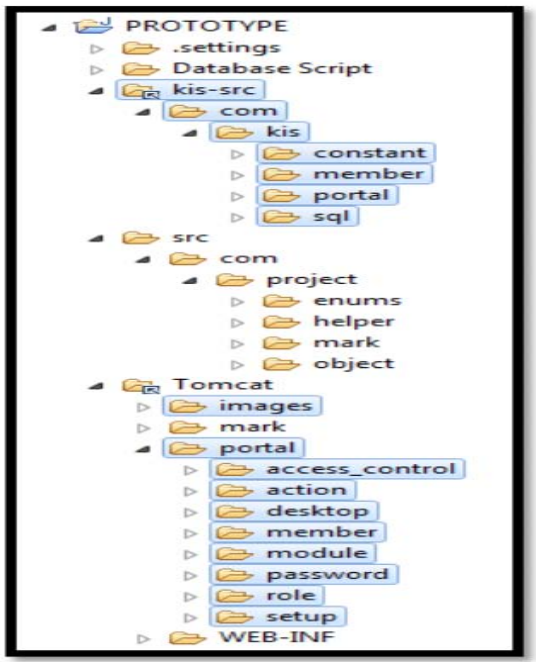

Fig. 2: Development project directory structure

Prototyping iteration I: In the first design and construction iteration, a single module is developed with all basic required functionality to perform key-in mark activities. The information is gathered from the software unit of the organization and multiple resources from the internet. Based on the collected data in the design phase, a flowchart is constructed to address the module process or operational flow. Then followed by the entity relational diagram for clear and organized database construction. Before the module ready to be developed, the development environment and specifications is essential to be setup. The list of components required for the development.

Development componemts:

- Windows 7 profesional-operational system
- Java Development Kit (JDK) 6 update 43-Java progamming language
- Eclipse kepler-Java Integrated Development Enviroment (IDE)
- Apache tomcat 6.035-web server
- My SQL 5.5 datebase

The development of a key-in mark module took several days including functionality test. The development begins with the configuration of Keep It Simple (KIS framework) by importing KIS project directory and libraries into Eclipse IDE. Each individual module then being registered in the KIS portal. Those directories is highlighted and shown in Fig. 2. Basically, one of research objective is to develop a prototype web application. The prototype is developed using Keep It Simple (KIS) portal, a rapid development framework. KIS is a modular-based architecture portal which means every functionality or feature in KIS is developed in a module.

The next step is a development of key-in mark module by constructing all required components as explained in previous section. The module is developed based on mvc (model, view, controller) architecture. The model consists of all classes in src/com/project/object and src/com/project/helper directory; meanwhile view consists of all jsp files in/tomcat/mark directory and controller consists of all classes in src/com/project/mark directory. The src/com/project/object directory contains numbers of classes that represent each respective tables for example student java. Meanwhile, src/com/project/helper directory contains numbers of classes which responsible to either retrieve or manipulate the data by communicate to database. Example of the classes is student helper java.

The src/com/project/mark directory contains several classes which act as controller and responsible to control the interaction between model and view. The directory also contain a servlet class which is the hearth of the module named keyinmarkservlet.java. For the purpose of this module, there are only three controller classes required that are get key in mark list action. Java, edit key in mark action. Java and update key in mark action java. All of these classes was than registered in the web.xml file for them to be accessible from ui. The basic input validation is being control in these files.

The key-in mark module perform two actions that are to list all students with summarize marks and to update individual student's mark. Thus, it has two JSP files in tomcat/mark directory that arelist.jsp (student list page) and edit.jsp (update mark page) to serve as User Interface (UI).

Prototyping iteration II: Most developers will not be able to master application security domain, especially if they are busy learning the latest framework or programming language. In order to help organizations overcome this barrier, OWASP has identified and build a security API that covers all the security controls a typical enterprise web application or web service project might need. There are about 120 methods across all the different security controls, organized into a simple intuitive set of interfaces. OWASP Enterprise Security API (ESAPI) helps software developer's guard against security-related design and implementation flaws.

ESAPI is primarily a set of interfaces designed to make security easy to use. These interfaces are usable by

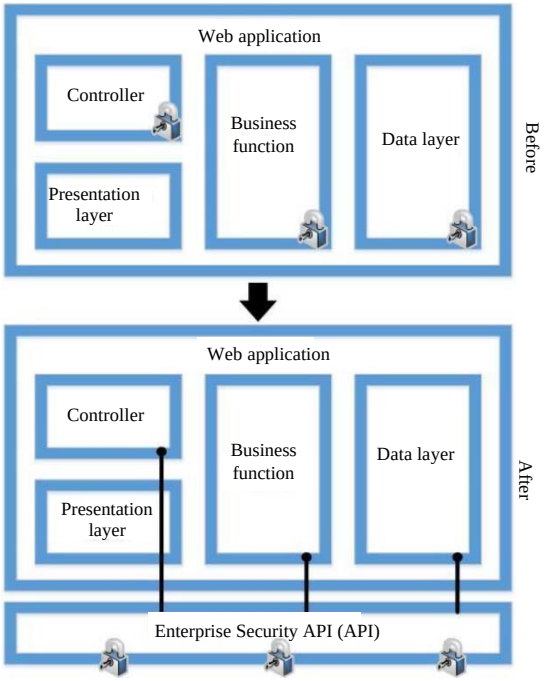

Fig. 3: ESAPI interface implementation

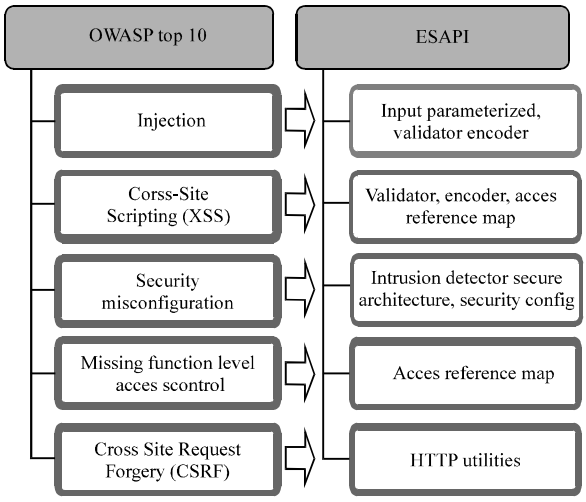

Fig. 4: Threats vs. ESAPI

a developer who cares to implement this security initiative for their application development. Figure 3 shows the Enterprise Security API (ESAPI) interface implementation in prototyping iteration II.

Enterprise Security API (ESAPI) covers wide range of application vulnerability and security control across OWASP top 10 list as discussed in the previous section. Figure 4 shows how the features of ESAPI match up against the selected five threats as listed in OWASP Top 10 project.

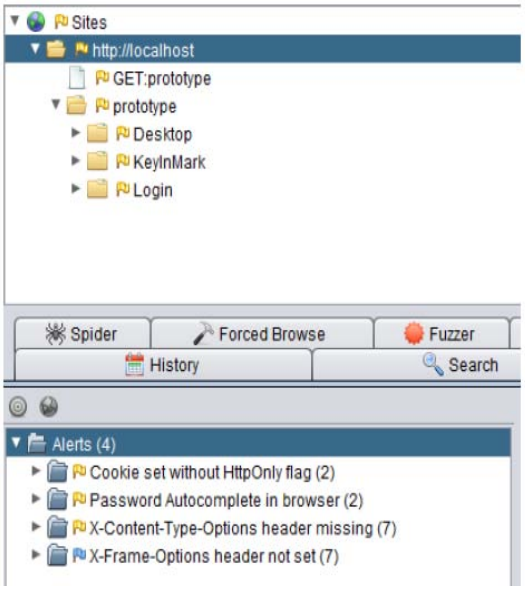

Fig. 5: Penetration test report

Before implementing ESAPI, first developers need to download the latest ESAPI distribution and unzip it into their application's lib directory. In this research, the latest API version found is esapi-2.1.0.jar. In addition to the latest API file, there are several libraries that are used by the ESAPI reference implementation that need to be included. These libraries might collide with libraries that are already exists in current application, thus developer need to be careful.

Next is to ensure that ESAPI resources are properly set up. The ESAPI reference implementation uses a 'resources/configuration' directory containing several files. This directory can be located anywhere on working environment classpath or can be specified with an environment variable on the Java command line as follows: -D org.owasp.esapi.resources = 'C:\resources'. The most important file in the resources directory is the ESAPI properties file. In this file, developer need to change the master password for the ESAPI installation. Various validation patterns, logging configurations, encryption algorithms and exception thresholds can be added.

Result based on ZAP analysis: They carried out application penetration test report as shown in Fig. 5 below. It shows only four type of alerts captured during active scanning executed. None of those alerts related to selected five serious application risks released by OWASP. This is proved that with the implementation of the Enterprise Security API (ESAPI) successfully mitigated five selected web application threats.

Based on the analyzed report produced by ZAP it is proven that the prototype application with ESAPI implementation are more resists against security threats and have better security handling compared to the other prototyped application without ESAPI implementation.

CONCLUSION

The study recommends that future study to concentrate on ESAPI implementation in other languages such as PHP, Python and.Net. The future study also might be more challenging due to new security threats and web application vulnerability found. The future study area advised to focus on various establishes application development framework or Content Management System (CMS) such as Liferay, Drupal and Joomla. The use of multiple security test tool may help to meet and validate the analysis result accurateness.

REFERENCES

Epstein, J., 2009. A survey of vendor software assurance practices. Proceedings of the Conference on 2009 Computer Security Applications Annual, December 7-11, 2009, IEEE, Virginia, USA., ISBN:978-1-4244-5327-6, pp: 528-537.

Fonseca, J., M. Vieira, K. Buragga and N. Zaman, 2013. A Survey on Secure Software Development Lifecycles. In: Software Development Techniques for Constructive Information Systems Design, Khalid, A.B and Z. Noor (Eds.). Idea Group Inc, Calgary, Alberta, pp: 57-73.

Kaur, D., P. Kaur and H. Singh, 2012. Secure spiral: A secure software development model. J. Software Eng., 6: 10-15.

Kothari, C.R., 2010. Research Methodology Methods and Techniques. New Age International Publishers, New Delhi, India.

Martin, D., 2011. Development and implementation of secure web application. Center for the Protection of National Infrastructure, UK.

Munassar, N.M.A. and A. Govardhan, 2010. A comparison between five models of software engineering. IJCSI, 5: 95-101.

Siddiqui, M.S. and S.J. Hussain, 2006. Comprehensive software development model. Proceedings of the IEEE International Conference on Computer Systems and Applications, March 8-8, 2006, IEEE, Virginia, USA., ISBN:1-4244-0211-5, pp: 353-360.