Int. J. of Comp. & Info. Tech., (2016) 4(2): 35-42.

IJOCIT

ISBN: 2345-3877
www.ijocit.org
Volume 4, Issue 2

Review Paper

Distinctive Vulnerabilities in Web Applications: A Literature Review

Er. Gurjot Singh [1*]
Rajdeep Kaur [1]
Arshdeep Kaur [1]

Received: 12 Feb 2016
Accepted: 07 May 2016

Abstract
Web applications are significant, common distributed systems whose current security relies primarily on server-side mechanisms. Web applications provide end users with client approach to server functionality through a set of web pages. These web pages often contain script code to be accomplished dynamically within the client web browser. Most web applications aim to impose simple, intuitive security policies, such as, for web-based email, disallowing any scripts in un-trusted email messages. Even so, Web applications are directly subject to a plethora of successful attacks, such as cross-site scripting, cookie theft, browser hijacking, and the new self-propagating worms in Web-based email and social networking sites. In this paper, we emphasized on the literature about web application vulnerabilities to eliminate common security exploits and to defend the emerging class of rich, cross-domain web applications.

Keywords: SQL Injection, Cross Site Scripting, Web Vulnerabilities.

Citation: Singh, E. G.; Kaur, R.; Kaur, A.(2016). Distinctive Vulnerabilities in Web Applications: A Literature Review, *Int. J. of Comp. & Info. Tech. (IJOCIT)*, 4(2): 35-42.

1 Department of Computer Science & Applications, K.M.V., (G.N.D.U.), Jalandhar, Punjab, India.
* Corresponding Author: dr.gurjotsingh@yahoo.com.

1. Introduction

With the development of information technology, [1] website security is very important because the website contain critical information about a company and now a day's website impairment is very common even a script kiddies and a new born hackers can do this. Despite their influences, web applications do raise a number of security involvements stemming from improper coding. Serious flaws or vulnerabilities, allow hackers to gain direct and public connection to databases in order to churn precise data. Many of these databases contain valuable information (e.g., personal and financial details) making them a numerous target of hackers. Although such acts of vandalism as obliterate corporate websites are still commonplace, nowadays, hackers prefer promoting access to the sensitive data consisting on the database server because of the enormous pay-offs in selling the data.

In the framework expressed above, it is easy to see how a hacker can instantly access the data consisting on the database through a dose of creativity and, with luck, failure or human error, prominent to vulnerabilities in the web applications.

As stated, websites depend on databases to distribute the required information to visitors. If web applications are not protect, i.e., vulnerable to, partially one of the distinct forms of hacking techniques, then your unified database of sensitive information is at serious risk. Some hackers, for example, may maliciously infuse code within vulnerable web applications to hoax users and redirect them against phishing sites.

This procedure is called Cross-Site Scripting and may be used even still the web servers and database engine consist of no vulnerability themselves. Recent research parades that 75% of cyber-attacks are done at web application level.

The remainder of this paper is organized as follow: in the section 1.1 we introduce the web applications and in section 1.2, we discuss the distinctive web application vulnerabilities. In the next section 2, we present the literature on web application and their vulnerabilities. At last, we conclude the paper.

2. WEB Applications

From a technical view-point, the web is an extremely programmable environment that allows mass customization through the immediate deployment of a large and distinct range of applications, to millions of global users. Two important components of a modern website are malleable web browsers and web applications; both available to all and sundry at no expense.

Web browsers are the software applications that avow users to fetch data and interact with content located on web pages within a website. Today's websites are a deep cry from the static text and graphics showcases of the early and mid-nineties: modern web pages allow illustrated dynamic content to be pulled down by users according to individual preferences and settings. Furthermore, web pages may also rush client-side scripts that "change" the Internet browser into an interface for such applications as web mail and associative mapping software (e.g. Yahoo Mail and Google Maps).

Most importantly, modern web sites concede the capture, processing, storage and transmission of sensitive customer data for immediate and repetitive use and, this is done through web applications. Such appearance as webmail, login pages, brace and product request forms, shopping carts and content executive systems, shape modern websites and provide businesses with the means crucial to correspond with prospects and customers. These are all familiar examples of web applications.

Web applications are, therefore, computer programs conceding website visitors to submit and retrieve data to/from a database over the Internet using their approved web browser. The data is then conferred to the user within their browser as information is achieved dynamically by the web application through a web server.

For the more technical oriented, Web applications query the content server (essentially a content repository database) and dynamically generate web documents to handle to the client (people surfing the website). The documents are generated in a standard format to allow brace by all browsers (e.g. HTML or XHTML). JavaScript is one form of client side script that permits dynamic elements on each page (e.g. image innovations). The web browser interprets and runs all scripts while exhibiting the requested pages and content.

Additional significant advantage of building and maintaining web applications is that they achieve their function irrespective of the operating system and browsers running on client side. Web applications are quickly set up anywhere at no cost and without any installation requirements at the user's end. As the number of businesses grasping the benefits of doing business over the web increases, so will the use of web applications and other related

technologies endure to grow. Moreover, since the increasing adoption of intranets and extranets, web applications become immensely entrenched in any organization's communication infrastructures, further broadening their scope and probability of technological complexity and prowess.

3. WEB Application Vulnerabilities

Application vulnerability is a system fault or weakness in an application that could be exploited to compromise the security of the application. Once an attacker has erected a flaw, or application vulnerability, and determined how to access it, the attacker has the probable to exploit the application vulnerability to facilitate a cyber-crime. These crimes affects the confidentiality, integrity, or availability (known as the "CIA triad") of resources possessed by an application, its designers and its users. Attackers typically rely on specific tools or methods to perform application vulnerability detection and compromise. According to Gartner Security, the application layer currently contains 90% of all vulnerabilities. The following list discloses some of the most common attacks that can be performed and each item will then be described in further details.

3.1. SQL Injection

SQL injection is a type of attacks that help application's vulnerabilities that interacts with a database, by injecting an unauthorized SQL query by an attacker in order to deal its security. SQLI is one of the most eminent and popular web applications vulnerability where the attackers exploits input vulnerability and attempts to send incorrect command or SQL query to the application [2].

3.2. Cross Site Scripting

Cross Site Scripting (XSS) ensues when an attacker is capable of injecting a script, usually JavaScript, into the output of a web application in such a way that it is executed in the client browser [3]. This generally happens by establishing a means of breaking out of a data text in HTML into a scripting text - usually by infusing new HTML, JavaScript strings or CSS markup. HTML has no lack of locations where executable JavaScript can be infused and browsers have even persuaded to add more. The injection is sent to the web application via any means of input such as HTTP parameters [3].

3.3. Invalidated Input

In web application the users are categorized in the distinct level of privileges. Access control determines how the web application avows access to functions to some users and not others, also called authorization. But attacker may be access higher level of authority. Web application desire inputs from user to determine how to react in accordance to serve web service.

3.4. Broken Access Control

The user invades input values to web application suit. Attackers may pass toxic information to the web application which tries to bypass the website's security mechanisms.

3.5. Broken Authentication and Sessions Management

In the web application when error ensues under normal activities and if not handled by appropriate error message to the user then there is opportunities of getting clues on flaws about the web application to the attackers and may disturb to the normal user. Web application creates session when the user logged in, which specify the duration of time that a rare user relates with a web application.

3.6. Improper Error Handling

Using session maintains state by contributing the client with a unique id. This id is gathered in a cookie which is used between the user browser and web server. If this session's details are not secured correctly, attacker can swipe it and misuse it.

3.7. Parameter Modification

Parameter modification is the problem where the attacker's do not padding the form but rather ravines the parameters from URL itself, bypassing the form validations. Hence it may lead to ambiguous effect on the form data and the overall site data.

3.8. Insecure Configuration Management

The web server that hosts the web application consists of web configuration files and directories which should not be earned or view by someone pirated. So must be preserved against the attackers.

3.9. Cookie Modification

Cookie stores the information in the text format which is used for case management. The web application needs cookies; the server delivers cookie and stores at client browser. The browser then returns the cookie to the server the next time the page is desired. The attackers can surely connect with the server to modify the contents of user's cookie [4].

3.10. Cross Site Request Forgery

This is a type of exploitation where attacker tricks victim into sending crafted web requests via image tag, XSS, or by means of other techniques with the victim's valid session identifier, however, on the attacker's behalf. Since the browser sends the sensitive information such as session cookies automatically, the victim's session being tempered, result in loss of sensitive information, also attacker can make forge requests that are undistinguishable from legitimate ones [5].

3.11. Command Execution

The specific purpose of the command injection attack is to inject and execute the commands that are specified by the attacker in the vulnerable application. These commands are executed with the same privileges as the application or web servers.

4. Literature Survey

Zoran Djuric [6] has purposed a tool named SQLIVDT that is designed for efficient SQLI vulnerability detection. The main goal of that tool is to generate test inputs & assess test results. Web application vulnerabilities allow attackers to perform malicious actions from unauthorized CCOUNT ACCESS. In last decade web application vulnerabilities are growing. The black box approach is based on simulation of SQL attacks against web applications.

Jose Fonseca, Nuno Seixas, Marco Vieira, and Henrique Madeira [7] had presented an analysis of 715 vulnerabilities, 121 exploits of 17 web applications. Some web applications were written in weak typed language and other in strong. According to paper weak typed are the performed targets. Most web application have critical bugs affecting their security .To prevent security problems it is important to understand their typical software faults. SQL Injection & XSS are two most widely spread and critical web application vulnerability. To understand how these vulnerabilities are exploited by hacker's paper presents an analysis of the source code of script used.

Jose Fonseca and Marco Vieira [8] have studied the use of static code analysis tool to detect vulnerabilities in the plugins. Results showed that many plugins that are currently deployed worldwide have dangerous cross site scripting & SQL Injection vulnerabilities that can be exploited. This paper analyzed the security vulnerabilities of 35 word press plugins using RIPS and PHPSAFE. More than 350 XSS & SOL unknown vulnerabilities were detected plugins are potential source of security problems. Effectiveness of static analysis tools needs to be improved, both in term of coverage and false positive. Many web applications allow the integration of 3rd part server side plugins offer diverse functionality and open an additional vulnerabilities door.

Mukesh Kumar Gupta, M.e.Govil and Girdhari Singh [9] have proposed a classification of software security approaches used to develop secure software in various phase of software development life cycle. Static analysis approaches are able to find out the cause of a security problem and can find errors. Error finding not only reduce the cost of error, also quick feedback cycle improves the coding approach. Static analysis approach suffers from false positive and false negative results. Dependence on web application is increasing very rapidly in recent time and create problem and for other purposes. Sol Injection and XSS are the most dangerous security vulnerability exploited in web application i.e. eBay, google, fb, etc. Most developer's repeat same programming mistake in their code because they do not follow security guidelines.

Adnan Masood and Jim Java [10] have presented the challenges of existing standards and reviews new techniques & tools to improve services security by detecting vulnerabilities are discussed. RESTFUL services have now become the new services development paradigm normal. The overview of the OWASP top 10 vulnerabilities for web services the potential static code analysis technique to discover vulnerabilities are discussed. Application layer vulnerability in the web services are a microcosm of the exploits available at the disposal of web application attackers. An iterative threat modeling approaches scan better prepare us to face and mitigate such attacks and make service oriented architecture as secure as possible.

Iberia Medeiros and Nuno Neves [11] have presented an approach for finding and correcting vulnerabilities in web applications and a tool to implement the approaches for PHP programs. Static source code analysis and data mining two techniques are used. The security of web application continues to be a challenging problem. In this paper the combination of method are used to discover vulnerabilities in source code with fewer false positives. WAP tools are used and large set of PHP applications are used for experimental evaluation. They found 388 vulnerabilities in 1.4 million lines of code.

J. Pradeep Kumar, Dr. T. Ravi and K. V. Nagendra [12] have presented a method for modeling

SOLIAs with the Augmented Attack TREE and regular expressions to capture subtle SQL statements formed by SQLIA adversaries. It also presents various security vulnerabilities in web application and provides the analysis for counter from security vulnerabilities. It shows various forms SOL Injection & Cross Site Scripting attacks.

Wang Chunlei, Liu Li and Liu Qiang [13] have presented a web services vulnerability identification and analysis method based on fuzz testing, including identifying inputs, generating fuzz testing data, performing fuzz testing, monitoring and identification of normal fragility, etc. Several vulnerabilities including SQL injection, information leakage, Xpath injection are discovered by using WSFuzzer to carry out web services vulnerability fuzz testing. It shows that the method proposed in this paper can effectively test the vulnerability of web services with high efficiency. When the experiments are conducted on the internet, firewalls and intrusion detection systems sometimes intercept the invocation of web service requests and have great impacts on fuzz testing.

Abdul Razzaq, Ali Hur, Sidra Shahbaz, Muddassar Masood and H Farooq Ahmad [14] have presented the compared of web application firewall solutions with important features necessary for the security at application layer. The available security solutions are ineffective due to their focus on network layer and limitations in their core technologies design. Various solutions available as open source and in commercial market are creating problem for selecting the suitable solution for the security of the organizational systems. Critical analysis on WAF solutions is helpful for the users to select the most suitable solution to their environment. Due to dramatic increase in web applications, security gets vulnerable to variety of threats. The basic reason behind success of these attacks is the ignorance of application developers while writing the web applications and the vulnerabilities in the existing technologies.

Marcelo Invert Palma Salas, Paulo Licio de Geus and Eliane Martins [15] have showed that web services have raised new challenges on information security; this technology is susceptible to XML Injection attacks, which would allow an attacker to collect and manipulate information to insert malicious code in either server side or client side, being one of the most employed attack against web applications. The fault injection technique improves the robustness of web applications, through the greater flexibility to modify the test cases and to find software bugs. The result shows that 82% of web services tested were vulnerable to XML Injection attacks.

Theodoor Scholte, William Robertson, Davide Balzarotti and Engin Kirda [16] have presented a novel technique for preventing the exploitation of XSS and SQL injection vulnerabilities based on automated a data type detection of input parameters. The author implemented IPAAS for PHP and evaluated it on five-world web application with known XSS and SQL injection vulnerabilities. The evaluation demonstrates that IPAAS would have prevented 83% of SQL injection and 65% of XSS vulnerabilities. Web application becomes an integral part of the daily lives of millions of users. But these applications are also targeted by attackers and critical vulnerabilities are still common. Current techniques focus mainly on sanitization: either on automated sanitization, the detection of missing sanitizers, and the correctness of sanitizers.

Chandershekhar Sharma and Dr. S.C. Jain [17] have presented Web application interacts with the back end database to retrieve data as and when requested by the user. SQL injection attacks are one of the top most threats in database centric web application and SQL injection vulnerabilities are the most serious vulnerability types. In this paper the classification of SQL injection attacks are discussed and also analysis is done on basis of risk associated with each attack. The result of analysis highlights the impact of attacks on the database of web applications. This analysis can be very helpful in designing the detection tools.

Seung-Jae Yoo and Jeong-Mo Yang [18] have proposed recently cyber hacking incidents and accidents increase significantly so that the damage has spread to businesses, society and national level. As web security is emerged as the most important security, it has emerged as core topics in security research. In this paper, Author builds a system to find unencrypted packets through real time packets monitoring and furthermore propose to build the applied encrypted system. In this, more than 3000pages are collected and analyzed including subpage of the homepage. Among them about 2100 pages were not encrypted. This approximately corresponds to a probability of 80%.

Yunhui Zheng and Xiangyu Zhang [19] have proposed a path and context sensitive inter procedural analysis to detect RCE vulnerabilities. This paper develop a prototype system and evaluate it on ten real world PHP applications, 21 true RCE vulnerabilities, with 8 unreported before are identified. A novel algorithm is developed to resolve the two sets of constraints together. The experiment shows that the technique is very effective in detecting RCE vulnerabilities in real-world PHP applications, producing much fewer false positives compared to alternative techniques. Remote code

execution attacks are one of the most prominent security threats for web applications. It is a special kind of cross site scripting attack that allows client inputs to be stored and executed as server side scripts.

Jan-Min Chen, and Chia-Lun Wu [20] have implemented an automated vulnerability scanner that for the injection attacks and scanned the injection attack vulnerabilities. They proposed mechanisms were a focus on SQL injection and Cross site scripting attack. As they according to detect based on injection point, so they can clearly know where the bug is, reduce the debug time and increase efficiency.

Ha Thanh Le and Peter Kok Keong Loh [21] have presented their experience with and experimental exemplification of using the Application Vulnerability Description Language (AVDL) to realize a unified data model for technology-independent vulnerability analysis of web applications. They conduct case studies with different web application scanners and evaluating their outputs using AVDL. They continue with description extraction tool and the unified data model as a data interface for the rule-based inference engine which incorporates vulnerability analysis and prediction capability.

José Fonseca, Marco Vieira and Henrique Madeira [22] have proposed an approach to evaluate and compare web application vulnerability scanners. They presented a method to evaluate and benchmark automatic web vulnerability scanners using software fault injection techniques. The percentage of false positives is very high, ranging from 20% to 77% in the experiments performed and the results show that in general the coverage is low.

Mattia Monga, Roberto Paleari and Emanuele Passerini [23] have presented an approach for detecting injection vulnerabilities in web applications through hybrid analysis techniques. They described the design and implementation of Phan, a hybrid analyzer for PHP applications that works directly at the Zendbytecode level. Their proposal blends together the strengths of static and dynamic approaches: the preliminary static analysis phase helps reducing the run-time overhead connected with dynamic monitoring. The preliminary results indicate that the improvement with respect to a taint analysis entirely dynamic is significant.

Xin Wang, Luhua Wang, Gengyu Wei, Dongmei Zhang and Yixian Yang [24] have proposed a SQL injection vulnerability detection method based on hidden web crawling. They combine authentication with the crawler model, and find SQL injection vulnerability by simulating web attacking and analyzing the data of response, also did two experiments, one is to compare the coverage of our tool with other three tradition scanners [10, l3, 24] by detecting three common public web sites, and the result shows that the system we implemented can retrieve hidden web pages and its page coverage is larger than other three scanners.

Gang Zhao and Hua Chen [25] have presented the use of data-flow based methods to analyze the vulnerability of Java program in bytecode. Data-flow analysis is a technique to expose the global influence of data in a program. They discussed the characteristics of the software vulnerability. The relationship of the vulnerability analysis and data-flow analysis is investigated. There is a suggestion about the framework of a data-flow based analysis system. Its implementation aims to Java bytecode is brought out.

Weider D., Yu Shruti Nargundkar and Nagapriya Tiruthani [26] have proposed in detail the various methods used in phishing. They perform a root-cause analysis of the methods used in phishing, the motivation for phishing and in the process come up with a fishbone diagram outlining the causes and methodologies used in phishing. This analysis is aimed at helping developers to design and develop better anti phishing solutions.

Lwin Khin Shar, Lionel C. Briand and Hee Beng Kuan Tan [27] have proposed using a set of hybrid (static+dynamic) code attributes that characterize input validation and input sanitization code patterns and are expected to be significant indicators of web application vulnerabilities. Because static and dynamic program analyses complement each other, both techniques are used to extract the proposed attributes in an accurate and scalable way. Therefore, building predictors using machine learners trained with the information provided by both static and dynamic analyses and available vulnerability information, they achieve good accuracy while meeting scalability requirements.

Tânia Basso, Plínio César Simões Fernandes and Mario Jinoand Regina Moraes [28] have presented an experimental study where they analyzed the effect that Java software faults, injected on the source code of Web applications, can have on security vulnerabilities. Also, they analyzed the influence of these faults on the security vulnerabilities detection by a well-known commercial web security vulnerability scanner tool. The results of the scanner tool were validated through manual attacks based on attack trees.

Nuno Antunes and Marco Vieira [29] have presented an experimental study on the comparison of several web vulnerability detection tools implementing either penetration-testing or static code. They showed how effective these two techniques are on the detection of SQL Injection vulnerabilities in web services code. The Results suggest that, in general, static code analyzers are able to detect more SQL Injection vulnerabilities than penetration testing tools and coverage of static code analysis tools is typically much higher than of penetration testing tools.

5. Conclusion

In this paper, we mainly emphasized on discussing the literature about the destructive vulnerabilities of web applications as these web applications have more significantly popular and have wide spread interaction medium in the present era. But at same point these destructive vulnerabilities disclosed the user`s sensitive information regularly. These vulnerabilities should be analyzed efficiently and proper countermeasures are implemented to prevent the user`s critical information from being tampered.

6. References

[1] http://www.acunetix.com/websitesecurity/web-applications/

[2] Nadya ElBachir El Moussaid, Ahmed Toumanari, (2014), "Web Application Attacks Detection: A Survey and Classification", International Journal of Computer Applications Volume 103, No.12.

[3] Http://phpsecurity.readthedocs.org/en/latest/Cross-Site-Scripting-(XSS).html. [Accessed on February 2016]

[4] Katkar Anjali S., Kulkarni Raj B, (2012), "Web Vulnerability Detection and Security Mechanism", International Journal of Soft Computing and Engineering, Volume-2, Issue-4.

[5] Gopal R. Chaudhari, Prof. Madhav V. Vaidya, (2014), "A Survey on Security and Vulnerabilities of Web Application", International Journal of Computer Science and Information Technologies, Vol. 5 (2).

[6] Zoran Djuric, (2013), "A Black-box Testing Tool for Detecting SQL Injection Vulnerabilities", IEEE Second International Conference on Informatics and Applications, pp. 216- 221.

[7] Jose´ Fonseca, Nuno Seixas, Marco Vieira, and Henrique Madeira, (April 2014), "Analysis of Field Data on Web Security Vulnerabilities", IEEE transactions on dependable and secure computing, Vol. 11, No. 2, pp. 89- 100.

[8] José Fonseca and Marco Vieira, (2014), "A Practical Experience on the Impact of Plugins in Web Security", IEEE 33rd International Symposium on Reliable Distributed Systems, pp. 21-30.

[9] Mukesh Kumar Gupta, M.E. Govil and Girdhari Singh, (May 2014), "Static Analysis Approaches to Detect SQL Injection and Cross Site Scripting Vulnerabilities in Web Applications", IEEE International Conference on Recent Advances and Innovations in Engineering, Jaipur, India.

[10] Adnan Masood, Jim Java, (2015), "Static Analysis for Web Service Security – Tools & Techniques for a Secure Development Life Cycle", International Symposium on Technologies for Homeland Security, pp. 1-6.

[11] Ibéria Medeiros, Nuno Neves, (2015), "Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining", IEEE TRANSACTIONS ON RELIABILITY, pp.1-16.

[12] J. Pradeep Kumar, Dr. T. Ravi and K. V. Nagendra, (2012), "Analysis of security vulnerabilities for web based application" IEEE, pp. 233- 236.

[13] Wang Chunlei, Liu Li, Liu Qiang, (2014), "Automatic fuzz testing of web service vulnerability" International Conference on Information and Communications Technologies, pp. 1-6.

[14] Abdul Razzaq, Ali Hur, Sidra Shahbaz, MuddassarMasood, H Farooq Ahmad, (2013), "Critical Analysis on Web Application Firewall Solutions", IEEE Eleventh International Symposiumon Autonomous Decentralized Systems, pp. 1-6.

[15] Marcelo Invert Palma Salas, Paulo Lício de Geus, Eliane Martins, (2015), "Security Testing Methodology for Evaluation of Web Services Robustness - Case: XML Injection", IEEE World Congress on Services, pp. 303-310.

[16] Theodoor Scholte, William Robertson, Davide Balzarotti, EnginKirda, (2012), "Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis", IEEE 36th International Conference on Computer Software and Applications, pp. 233- 243.

[17] Chandershekhar, Dr. S.c. Jain, (2014), "Analysis and Classification of SQL Injection Vulnerabilities and Attacks on Web Applications", IEEE International Conference on Advances in Engineering & Technology Research, pp. 1-6.

[18] Seung-Jae Yoo, Jeong-Mo Yang, (2014), "Web login Vulnerability Analysis and Countermeasures", International Conference on IT Convergence and Security, pp. 1-4.

[19] Yunhui Zheng and Xiangyu Zhang, (2013), "Path Sensitive Static Analysis of Web Applications for Remote Code Execution Vulnerability Detection", IEEE 35th International Conference on Software Engineering, pp. 652- 661.

[20] Jan-Min Chen, and Chia-Lun Wu, (2010), "An Automated Vulnerability Scanner for Injection Attack Based on Injection Point", International Computer Symposium (ICS), pp. 113- 118.

[21] Ha Thanh Le and Peter Kok Keong Loh, (2008), "Evaluating AVDL Descriptions for Web Application Vulnerability Analysis", ISI 2008, June 17-20, 2008, Taipei, Taiwan, IEEE, pp. 279-281.

[22] José Fonseca and Marco Vieira, Henrique Madeira, (2007), "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks", 13th IEEE International Symposium on Pacific Rim Dependable Computing, pp. 365-372.

[23] Mattia Monga, Roberto Paleari, Emanuele Passerini, (2009), "A hybrid analysis framework for detecting web application vulnerabilities", ICSE'09 Workshop, Vancouver, Canada, IEEE, pp. 25- 32.

[24] Xin Wang, Luhua Wang, Gengyu Wei, Dongmei Zhang, Yixian Yang, (2010), "Hidden web crawling for SQL injection detection", 3rd IEEE International Conference on Broadband Network and Multimedia Technology, pp. 14-18.

[25] Gang Zhao and Hua Chen, (2008), "Data-flow Based Analysis of Java Bytecode Vulnerability", The Ninth International Conference on Web-Age Information Management, pp. 647- 653.

[26] Weider D., Yu Shruti Nargundkar and Nagapriya Tiruthani, (2008), "A Phishing Vulnerability Analysis of Web Based Systems", IEEE Symposium on Computers and Communications, 2008. Pp. 326- 331.

[27] Lwin Khin Shar, Lionel C. Briand and Hee Beng Kuan Tan, (2013), "Web Application Vulnerability Prediction using Hybrid Program Analysis and Machine Learning", IEEE 36th International Conference on Computer Software and Applications, pp. 1-35.

[28] Tânia Basso, Plínio César Simões Fernandes, Mario Jinoand Regina Moraes, (2010), "Analysis of the Effect of Java Software Faults on Security Vulnerabilities and Their Detection by Commercial Web Vulnerability Scanner Tool", International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 150- 155.

[29] Nuno Antunes, Marco Vieira, (2009), "Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services", 15th IEEE Pacific Rim International Symposium on Dependable Computing, pp. 301- 306.