American Journal of Computing Research Repository, 2016, Vol. 4, No. 1, 15-20
Available online at http://pubs.sciepub.com/ajcrr/4/1/3
© Science and Education Publishing
DOI:10.12691/ajcrr-4-1-3

A Hybrid Algorithm for Detecting Web Based Applications Vulnerabilities

Muiruri Chris Karumba*, Samuel Ruhiu, Christopher A. Moturi

School of Computing and Informatics, University of Nairobi, Nairobi, Kenya
*Corresponding author: chrismuiruri@yahoo.co.uk

Abstract Web vulnerability scanners (WVS) are tools for discovering vulnerabilities in a web application. However, they are not 100% accurate. In this paper we develop a hybrid algorithm for detecting web based applications vulnerabilities and compare its performance with other open source WVS. The comparison is based on three metrics namely time taken to scan, detection accuracy and consistency.

Keywords: *web vulnerability scanners, open source, algorithm, web based applications*

Cite This Article: Muiruri Chris Karumba, Samuel Ruhiu, and Christopher A. Moturi, "A Hybrid Algorithm for Detecting Web Based Applications Vulnerabilities." *American Journal of Computing Research Repository*, vol. 4, no. 1 (2016): 15-20. doi: 10.12691/ajcrr-4-1-3.

1. Introduction

The number and importance of web applications have increased rapidly over the year [11] many organizations have embraced these technologies to explore new business opportunities and some companies have been forced to adopt the electronic commerce by their customers or competitors.

Web applications have gained popularity and have become part of our daily lives interaction. New web application vulnerabilities emerge every now and then and endanger the use of the web-based applications. Therefore, manual code inspection or security audits must be done by highly trained experts who are labour-intensive, expensive, and prone to errors. For this reason, there is need to automate vulnerability discovery.

Vulnerabilities exist in web applications due to development of such application by less experience programmers or failure to test these applications for loopholes. This results in deployment of vulnerable web application that can be hacked exposing confidential information.

1.1. Problem Statement

With advancement in web technologies and shift from traditional desktop application to web-based solutions, the popularity of web-based applications has grown tremendously. Today, the web-based applications are used in security-critical environments, such as medical, financial and military systems [28]. Although the internet infrastructure is developed by experienced programmers with security concerns in their mind, some of the web applications are engineered by less experienced consulting programmers with little or no knowledge about security. This exposes the web application to various vulnerabilities and provides avenues for cyber criminals to gain unauthorized access to confidential information.

In one of the recent studies by the Ponemon Institute, they found out that that 45% of breaches exceed $500,000 in losses. In the largest of incidents, many Fortune-listed companies have given shareholder guidance that the losses would vary from a few dollars to millions of dollars. For this reason, it is prudent to do something in a proactive manner to avert or reduce harm before a cyber-attack.

Past studies have concentrated in benchmarking open source web vulnerability scanners to find out their capabilities and limitations. There is need to analyze different algorithms, identify their strength and weaknesses, with an aim of coming up with a hybrid algorithm that is superior, performs faster and can work on more inputs and in a complex situation. In this proposal, the researcher seeks to benchmarks different web vulnerability scanners, identify these tools and suggest an improved algorithm that can be adopted while developing tools.

2. Literature Review

According to [26] many organizations rely upon customized web applications to implement business processes. These may include full-blown applications, or consist of modules such as online, login pages shopping carts, and other kinds of dynamic content. Some of these software applications in your network could be developed in-house

In their study, [31] noted that web application security vulnerabilities are on the increase. These vulnerabilities allow attackers to perform undesirable actions that range from gaining unauthorized account access, to obtaining confidential data such as credit card numbers and in some extreme cases, they threaten to reveal the identitiesof intelligencepersonnel.

Table 1. Web applicationVulnerabilities

Web applicationVulnerabilities	
-Remote file inclusion	-Command Injection
-Local file inclusion,	-Blind SQL Injection
-Cross site crossing,	-SQL Injection,
-Cross site scripting,	-LDAP Injection,
-Sessionmanagement,	-Buffer overflow
-sever side injection	-X-path Injection
-cross site refrence forgery	-HTTP Splitting

3. Methodology

3.1. Research Design

A hybrid algorithm was designed based on the logic expressed in the diagram below. The major milestone of this algorithm is to increase the detection accuracy.

The hybrid algorithm is derived from existing algorithms with a goal of increasing the vulnerability scanning accuracy and the time it takes to scan any given web application. Although the accuracy may not be achieved 100% emphasis, an effort has been put to raise it above the existing tools. The results of the tests will be benchmarked with OWASP results which is updated on a regular basis.

The hybrid algorithm is based on the concept of carefully, combining desirable features of various components so that the new algorithm has the ability to discover more vulnerabilities. However the combination is not done blindly, it is based on various factors such as optimization and sophistication among others with an aim of increasing efficiency.

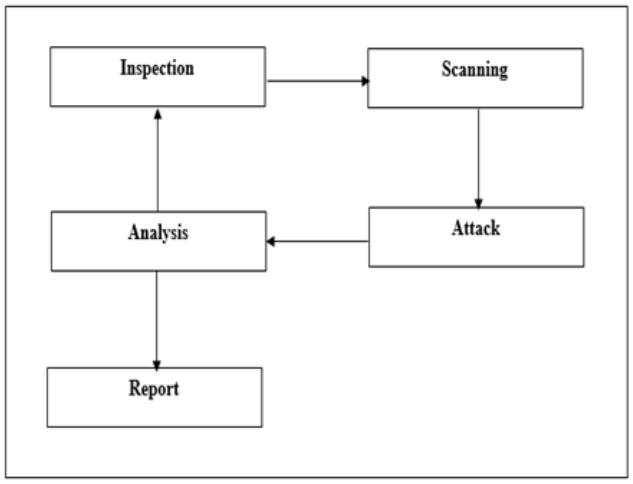

Figure 1. Hybrid Algorithm methodology

The hybrid algorithm comprises of five phases as shown in the figure above. Inspection or crawling this phase focuses on looking for information about the web application. The more the details found on this stage, the more successful the entire scanning process will be. Once the first phase is completed, the Scanning process begins, which involves, recognizing the weaknesses that exist in the web application. Once the vulnerabilities are discovered they are analysed in the next phase and then a report is displayed at the end of the entire process.

3.2. Simulation Design

A program to test and validate the hybrid algorithm is built, based on the flowchart below. This simulation will be useful in testing and validating the hybrid algorithm. The user will input the URL (uniform resource locator) of the web application to be scanned and click on the scan button.

The scanning process involves crawling and parsing and the discovery of the vulnerabilities, this process is repeated until all the vulnerabilities have been discovered. Once this process is completed, the analysis is done and finally a report is displayed showing the discovered vulnerabilities discovered and their location.

The scanning process includes, crawling and fuzzing. After the scanning process is completed, the results are submitted for analysis and a report is displayed.

Input: The URL of the web application to be tested. This is provided by the user who initiates the web scanning process.

Processing: This involves crawling all the web pages, fuzzing, and identification of any weakness and firing inputs to check for any vulnerability.

Output: The results of processing are analysed and presented in a report format.

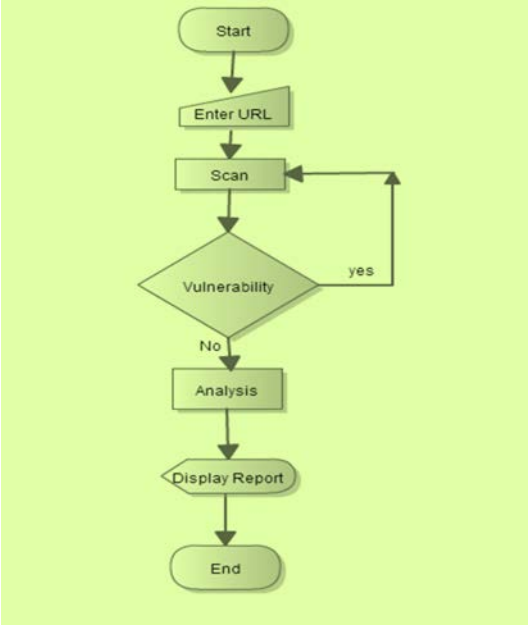

Figure 2.

3.3. SQL Injections Discovery

The scanning method described in the algorithm below is used as a method two for checking SQL. Check Vulnerability (V) which looks for any special characters, and Boolean characters and keywords in the input fields of a web based application. It has a compilation of all the special characters such as <=>{([',&+=<>=])} and a comprehensive collection of major keywords such as update, select, intersect, union insert, delete, drop, truncate and Boolean characters such as , 'AND' 'or '|'or','.

Using the actual inputs like a user interacting with a web browser, the values are tested against the database. If a mismatch is found in the results are submitted to the vulnerability information collector and then resets the Http request.

The algorithm below detects SQLIs in an effective manner. Which can be applied for any real web-based applications wherever the user and the database interacts

3.4. SQLAlgorithm

1. initialize sql characters in an array
2. create two maps or lists to store the sql error messages
   i. one for storing specific database error messages like oracle, mysql, microsoft sql error messages etc
   ii. Other for storing generic database error messages
3. Initialize error values in to the maps/list mentioned above
4. Initialize the scanner method – the scanner accepts the http message as input from the the crawler - http message has details on each request or url with the parameter list
5. For each parameter in the http message
   i. Input sql characters from the sql characters array
   ii. Verify the response to check for any matches on error messages from the two maps or lists
   iii. If a match occurs -Flag as sql vulnerability
   iv. Else - Repeat step 5 until the end of parameter list is reached
6. End

Flowchart

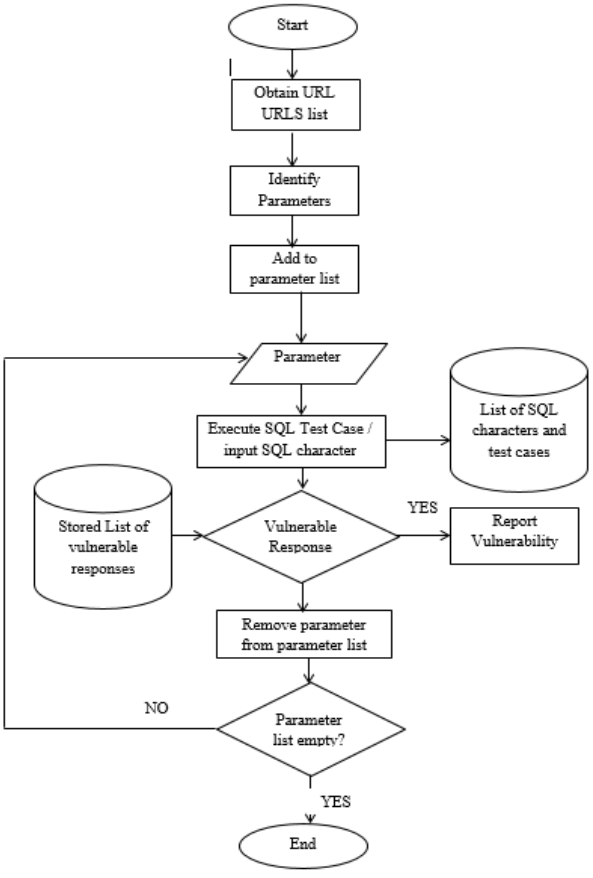

Figure 3.

Cross Site Scripting Algorithm

1. For each url in the list of visited urls
   a. Identify all parameters
   b. Push parameters in to parameter list
   c. For each parameter in the parameter queue
      i. Supply a **script** or a ***XSS test case*** as input to the parameter and pass the request
      ii. Verify the response to identify the supplied script or test case reflected back
2. Report the vulnerability if the response has a script

Flowchart

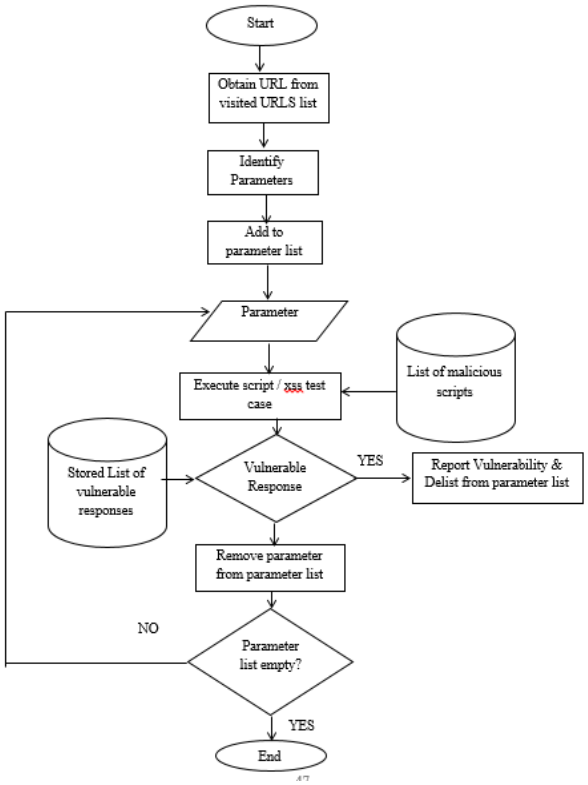

Figure 4.

3.5. Data Collection and Analysis

Data collection was done by using experiments. The data collected was further analysed into tables, graphs and pie charts.

3.6. Experimental Design

The experiments were performed by running seven opens source web application scanners on four web applications with known vulnerabilities. These web applications were installed on virtual machines which have similar configurations and resources.

3.7. Resources Required for the Experiment

i. Computer configured with four virtual machines
ii. Operating system: windows 8.1 professional edition
iii. Hypervisor such as VMware
iv. Web servers such as apache, tom cat and Xamp.

These web applications were installed on virtual machines which had similar configurations and resources.

The virtual machines specifications are processor, 2.6 GHZ Core i5, 2 GB RAM, 100 GB HDD and running on windows 8.1 professional edition.

In an effort to make sure that we have a similar test environment, similar configurations were used on the virtual machines regardless of the test conducted. Each scanning tool was run against identical yet distinct environment. This was critical to ensure that we obtain actual results without deviation due to different resources.

Table 2. Vulnerability scanners and web applications

Open Source Web scanning tools	Web applications with known vulnerabilities
• Wapiti • Websecurify • Arachni • W3af • Zed Attack Proxy • Vega	• OWASP WebGoat • Mutillidae • zero.weapplication.com • phpBB

Ck AppScan – A tool that was developed by the researcher to test and validate the hybrid algorithm.

3.8. Testing Procedure

In this research, all the selected web scanning tools were run on the aforementioned web applications and the results recorded. Below find the testing procedure.

i. Launch the web scanning tool
ii. Enter the url of the web application to be tested
iii. Click on the scan button and wait for the scanning process to be completed
iv. If the scan is carried out successfully, a report will is displayed with the results. During the scanning process web vulnerabilities discussed in section 2.3 are scanned and if found they will be displayed in the report.

This process was repeated for all the tools used in this research project.

Testing the Algorithm

The algorithm was be tested by translating it in to a simulation developed using java platform. The simulation was be run against the four web applications and the results collected about detection accuracy, the time taken to scan a given application as well as reliability and consistency. After the testing process the results of the simulation were compared with the other opens source web scanners.

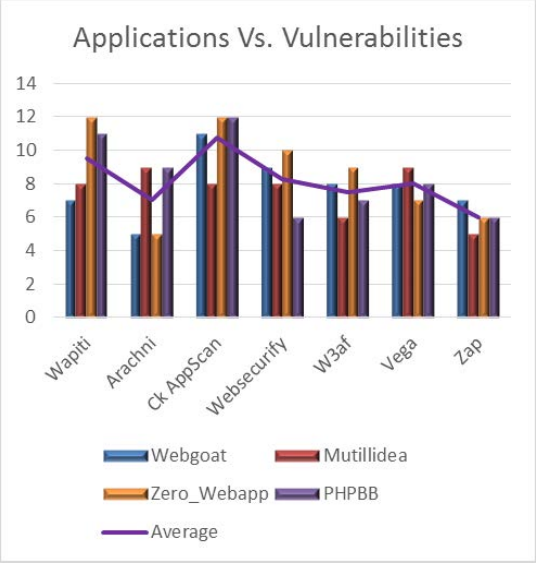

Figure 5. Web Scanning Applications Vs. Vulnerabilities

4. Results and Discussion

4.1. Results

Ck AppScan generated better results overall when compared to other tools used, although the tool has higher detection accuracy when compared to the other tool. The only drawback is that it was reported to take a longer time to scan than most of the web scanners that were used in this study. Its performance is not 100% accurate but it has a higher capacity to detect more vulnerabilities when compared to the other tools.

4.2. Discussion

A comparative study of web vulnerability scanners has also been performed by other researchers from different parts of the world. Although the tools and web applications used are not similar, the vulnerabilities are the same.

In a study conducted by Doupe et al (2010) they used Acunetix WVS, burp scanner, IBM's rational app scan, hailstorm, N-stalker, mileScan, Grendel-scan, NTO spider, W3AF, and HP web inspect against a web application known as wackPicko. They tested vulnerabilities such as XSS, SQL injection, file inclusion, file exposure and command line injection. The conclusion of the study was similar to those drawn by this study. They found out that crawling modern web based applications is indeed a serious challenge for many WVSs. There should be improved and more sophisticated algorithms needed to perform deep crawling. During the development of the hybrid algorithm the researcher was able to develop an algorithm that was able to detect the aforementioned vulnerabilities. This was achieved by employing a sophisticated method of discovering the weaknesses.

In a study conducted by Fonseca et al (2014) shows that many open source WVS have a low ability to detect vulnerability. This is in line with the results analysed after the end of this study. The researcher has developed a more sophisticated algorithm that address this concern and increased the number of vulnerabilities detected.

Khoury (2011) analysed three state-of–art black box WVSs against stored SQLI, and their results showed that stored (persistent) SQLI are not detected. The researcher was able to detect persistent SQL injections by fuzzing web applications using complex discovery algorithms.

Shelly (2010) performed a similar study by using several penetration tools to analyse the performance of several WVS. She used a mix of commercial and open source tools such as wapiti, Grendel-scan, Acunetix WVS, N-stalker, W3AF, and hailstorm. These tools were run against a modified version of BuggyBank web based application. They tools were tested for SQLI, XSS, buffer overflow and session management. The conclusion of this study was that the testing of WVS using secure and non-secure applications is indeed a suitable method to discover web vulnerabilities. In addition, she reported that for the discovery of non-traditional instances of XSS, SQLI, buffer overflow, malicious file execution and session management flows, more research needs to be done to improve the detection mechanisms used by these tools. The researcher addressed this issue by use of advanced heuristics and permutations during the detection process.

The hybrid algorithm was able to address concerns raised by previous researchers in different studies. This was achieved by adoption of more than one method during the vulnerability discovery process as well as improvement of the existing vulnerability detection methods. For instance it was noted that most of the WVS use either GET or POST method to detect weaknesses. The use of the two methods requires more scanning time nevertheless, more accurate results are realised.

5. Conclusion and Recommendations

5.1. Conclusion

The proposed hybrid algorithm is extensive in the execution of its detection mechanism against web application vulnerabilities. Testing of web applications for weaknesses is a significant step in safeguarding web applications. The proposed hybrid algorithm reports more vulnerabilities and presents a proficient manner while reporting discovered vulnerabilities. However since the proposed hybrid algorithm did not scan 100% of the existing vulnerabilities. There is need to increase the algorithm crawling component in order to ensure that it executed "deep" crawling. In addition the results presented shows that the proposed algorithm needs to be optimised to do the scanning in a short period of time. More research is needed to come up with a sophisticated algorithm that has the capacity to detect more vulnerabilities.

5.2. Recommendations for future work

i. Improved analysis and reporting

The hybrid algorithm needs an improved analysis and reporting structure. The tools need to analyze the application in a more efficient manner and deliver the desired results.

ii. Improved crawling capabilities

The hybrid algorithm requires a more sophisticated crawling mechanism, it should be engineered further to ensure that all the contents of a web application are scanned without omitting anything.

iii. Improved fuzzing component

There is a need to use a more advanced logic in the fuzzing component of the algorithm to get getter results. The fuzzing component is responsible for firing the necessary inputs to determine whether vulnerabilities exist or not. The fuzzing logic used by Ck AppScan should be improved further to increase the detection accuracy.

iv. Reduced scanning time

Some of the tools used in this study took a very long period of time to scan a web application. For this reason, there is a need for tools that can scan web applications within a short time and provide accurate results. it is important to improve the overall scanning mechanisms of the algorithms.

Acknowledgement

I would like to sincerely thank the almighty God and the University of Nairobi who has been of great assistance during the research period.

References

[1] Alssir, F. T., & Ahmed, M. (2012). Web Security Testing Approaches: Comparison Framework. In Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science (pp. 163-169). Springer Berlin Heidelberg.

[2] Antunes & Vieira (2012). Defending against web application vulnerabilities. Computer, (2), 66-72.

[3] Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (2010). State of the art: Automated black-box web application vulnerability testing. In Security and Privacy (SP), 2010 IEEE Symposium on (pp. 332-345). IEEE.

[4] Chen, S. (2014). wavsep. Available: http://sectooladdict.blogspot.com/2014/02/wavsep-web-application-scanner.html. [Accessed 09 July 2015.]

[5] Dessiatnikoff, A., Akrout, R., Alata, E., Kaaniche, M., & Nicomette, V. (2011). A clustering approach for web vulnerabilities detection. InDependable Computing (PRDC), 2011 IEEE 17th Pacific Rim International Symposium on (pp. 194-203). IEEE.

[6] Dougherty, C. (2012).Practical Identification of SQL Injection Vulnerabilities. 2012. US-CERT-United States Computer Emergency Readiness Team. Citado na, 34. [Accessed: 08th June 2015].

[7] Doupe, A., Cova, M., & Vigna, G. (2010). Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. In Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 111-131). Springer Berlin Heidelberg. [Accessed: 10th June 2015].

[8] Fonseca, J., Vieira, M., & Madeira, H. (2014). Evaluation of Web Security Mechanisms using Vulnerability & Attack Injection. Dependable and Secure Computing, IEEE Transactions on, 11(5), 440-453.

[9] Granville, K . (2015).Nine Recent Cyber-attacks against Big Businesses. New York Times [online] Available from: http://www.nytimes.com/interactive/2015/02/05/technology/recent-cyberattacks.html?_r=1. [Accessed 08 July 2015.].

[10] Howard, M., LeBlanc, D., & Viega, J. (2010). 24 deadly sins of software security [electronic book]: Programming flaws and how to fix them. New York: McGraw-Hill.

[11] Jovanovic, N., Kruegel, C., & Pixy, E. K. (2010). A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short Paper). In Proceedings of the 2006 IEEE symposium on Security and Privacy, Washington, DC, IEEE Computer Society (pp. 258-263).

[12] Kalman., G. (2014). Ten Most Common Web Security Vulnerabilities.[online] Available from: http://www.toptal.com/security/10-most-common-web-security-vulnerabilities [Accessed 08 July 2015.]

[13] Kals, S., Kirda, E., Kruegel, C., & Jovanovic, N. (2014). A web vulnerability scanner. In Proceedings of the 15th international conference on World Wide Web (pp. 247-256). ACM.

[14] Khoury, N., Zavarsky, P., Lindskog, D., & Ruhl, R. (2011). Testing and assessing web vulnerability scanners for persistent SQL injection attacks. In Proceedings of the First International Workshop on Security and Privacy Preserving in e-Societies (pp. 12-18). ACM.

[15] Kothari, C. R. (2009). Quantitative Techniques, 3E. Vikas publishing house PVT LTD.

[16] McQuade, K. (2014). Open Source Web Vulnerability Scanners: The Cost Effective Choice?. In Proceedings of the Conference for Information Systems Applied Research ISSN (Vol. 2167, p. 1508). [Accessed: 18th June 2015].

[17] Mirjalili, M., Nowroozi, A., & Alidoosti, M. (2014). A survey on web penetration test.

[18] Mugenda, O. Mugenda (2009) Research Methods: Quantitative and Qualitative Approaches. Nairobi: ACTS.

[19] Myers, G. J., Sandler, C., & Badgett, T. (2011). The art of software testing. John Wiley & Sons.

[20] Nagpal, B., Chauhan, N., & Singh, N. (2015). Defending Against Remote File Inclusion Attacks on Web Applications. i-Manager's Journal on Information Technology, 4(3), 25.

[21] Park, N. (2015). Detection Experimentation and Validation of Web Applications using Both Static and Dynamic Analysis. International Information Institute (Tokyo). Information, 18(5 (A)), 1735.

[22] Tripathi, A., & Singh, U. K. (2011). On prioritization of vulnerability categories based on CVSS scores. In Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on (pp. 692-697).

[23] Saunders, M. N., Saunders, M., Lewis, P., & Thornhill, A. (2011). Research methods for business students, 5/e. Pearson Education India.

[24] Sekaran, U. (2011). Research methods for business: A skill building approach. John Wiley & Sons.

[25] Shelly, D.A. (2010) .Using a Web Server Test Bed to Analyse the Limitations of Web Application Vulnerability Scanners. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia. [Accessed: 10th June 2015].

[26] Shema. M, (2011). Web Application Security for Dummies. England: John Wiley & Sons Ltd. P27-68.

[27] Snyder, B. (2014). 5 huge cyber security breaches at companies you know. Available from: http://fortune.com/2014/10/03/5-huge-cybersecurity-breaches-at-big-companies/. [Accessed 08 July 2015.]

[28] Stuttard, D., & Pinto, M. (2011). The web application hacker's handbook: discovering and exploiting security flaws. John Wiley & Sons. Inc. p33-80, p200-243.

[29] Van der Loo, F. (2011). Comparison of penetration testing tools for web applications (Doctoral dissertation, Master thesis, Radboud University Nijmegen. http://www.ru.nl/publish/pages/578936/frank_van_der_loo_scriptie. pdf).[Accessed: 08th June 2015].

[30] WhiteHat Security team. (2015). WhiteHat Security Statistics Report 2015. Available From: https://www.whitehatsec.com/statistics-report/featured/2015/05/21/statsreport.html. [Accessed 09 July 2015.].

[31] Yu, Y., Yang, Y., Gu, J., & Shen, L. (2011). Analysis and suggestions for the security of web applications. In Computer Science and Network Technology (ICCSNT), 2011 International Conference on (Vol. 1, pp. 236-240).