Available Online at www.ijcsmc.com/journal

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 11, Issue. 2, February 2022, pg.89 – 98

A HYBRID APPROACH TO DETECT SECURITY VULNERABILITIES IN WEB APPLICATIONS

Tsitsi G. Mubaiwa[1]; M Mukosera[2]
[1]Department of Information Technology, Harare institute of technology, Harare, Zimbabwe
[2]Department of Information Technology, Harare institute of technology, Harare, Zimbabwe
[1] tgmubaiwa@gmail.com; [2] mmukosera@hit.ac.zw
DOI: https://doi.org/10.47760/ijcsmc.2022.v11i02.011

ABSTRACT:** The presence of security flaws allows deceitful operators to exploit web application weaknesses. The researcher brings a novel vulnerability assessment technique in this study that can enhance exposure detection rates while also improving efficiency by lowering the number of test results that reports the presence of a condition wrongly and tests result that implies the absence of a condition when it is actually present. The purpose of the experiment is on a cutting-edge tool that uses a hybrid method that combines white-box and black-box testing practices. The amalgamation in building the hybrid algorithm is not done blindly as it is based on extraordinary aspects like optimization and complexity amid others to make bigger effectivity. The algorithm viably identifies SQL injections, XSS injection and can be utilized in any genuine application that run on a web server, wherever the client and the database interrelates. Crawling and parsing to discover vulnerabilities are part of the scanning process.* **The process** ***is done repeatedly until all vulnerabilities have been discovered. A prototype was done to test and validate the hybrid method. Simulation was done using a tool developed in Python and the researcher included in this paper a comparison table and graph that pits the new scanner versus two other web - based scanners.
Keywords— Web application security, web vulnerabilities, SQL injection, cross-side scripting (XSS) Web Vulnerability Scanners.

I. INTRODUCTION

Vulnerability is a flaw in software connecting to web servers, that can be caused by a design failure or an execution fault, allowing the attacker to do harm to the application's stakeholders. The application's owner, users, and other entities are all stakeholders [1].

Vulnerabilities are flaws, bugs, or loopholes in an online application that can be leveraged by cybercriminals [2]. Web-based apps are the most widely used network-based option for providing ordinary services. It has revolutionized how common utilities can be accessed, and constructing modern web apps is now the preferred method. Client-side and server-side programming are being used to develop these services (apps). The client section employs a variety of software applications (.Net, PHP, Python, and Ruby). Asynchronous XML (AJAX) and JavaScript are typically used to connect these two sections using the HTTP or HTTPS connection. [3].

Because of their widespread availability, programs that run on a web server have become engrained in all and sundry's daily routine. The reason for this popularity is because they are generally free and internet-accessible characteristics. Web apps seem to be a main target for hackers as a consequence of their rising interest, [4]. Hackers are focusing their attention on widely used software usage, such as blog posts, social platforms, banking, and e-commerce, as well as their drawbacks. A vast network that connects computers all over the world is teeming with assailants that have a hostile and criminal intent who employ freshly devised attack routes to jeopardize the security of beginner users. Some do it for the excitement, while others have nefarious motives such as exploiting data to make quick money, theft, extortion, criminal activity, and so on [5]. As a result, ongoing determinations are needed to determine the intrusion procedure, goal, and resolutions for these freshly developed attacks.

As stated by the Open Web Application Security Project report 2021 [6], SQL injection and XSS are among the top ten security flaws in the majority of web apps. SQL injection allows a malicious user to conduct SQL queries without the system's permission. This attack is carried out by altering the interpretation of the underlying SQL query and injecting non-validated data with certain keywords, operators, or statements. Malicious code is executed in the database if the attack is successful.

The XSS vulnerability, often known as cross-site scripting, is a weakness permitting malicious programs to be sent to a web browser using JavaScript. It happens when a program that runs on a web server accepts user inputs via HTTP requests or files that haven't been validated properly. The attacker can acquire numerous session-ids and cookies, allowing him to mimic the prey by using a social engineering attack to deceive the prey into divulging personal private information.

II. LITERATURE REVIEW

A literature review was conducted to examine prevailing effort then ascertain research gaps in the subject of web application exposure assessments, as well as their limits and future directions. Of particular interest was a recent research paper done by Asra Kalim et al [8] which identified more than five research gaps in the field of web security. The next section summarizes related work and examines the current technologies and future scope of existing tools and methodologies.

A. Related work.

This research paper highlights four major detection/correction techniques that have been used by most researchers namely Taint analysis based technique, Runtime enforcement based technique, Testing based techniques and Prevention based techniques. The proposed tool in this research focuses on SQL injection and Cross-site scripting security susceptibilities.

The bulk of the pages in software that connect to web servers, that are susceptible to SQL injection outbreaks were detected using a network-based vulnerability scanner. The technology also generates a report that aids developers in addressing the flaws. The tool's false positives were also found to be limited, and the proportion of systems connected within a network determines the tool's efficiency. The scanner, on the other hand, has some high needs in terms of network bandwidth and system count. In addition, the tool's performance in a single-user setting was ordinary and underwhelming. As a result, rather than a single-user environment, the tool is best suited for a multi-client networked system [7].

Yau-Wen Huang [9], regarded software vulnerabilities as a problem of protected information flow and devised a method to prevent flaws from being accessed at runtime. As a result of the uncovering of security vulnerabilities as a problem with secure data transmission, a tool (WebSSARI) was created to prevent vulnerabilities from becoming exposed at runtime. The technology they designed is called WebSSARI (Web application Security by Static Analysis and Runtime Inspection). This instrument utilized a functional testing methodology based on lattices. Their technique and tool have a problem in that they just protect the vulnerability, and the programmer should manually repair the detected weaknesses.

O. Hallaraker and G. Vigna [10] suggested a methodology that circumvent cross-site scripting exposures. The researchers' strategy, displayed a reviewing framework designed for the JavaScript translator aimed at the Mozilla internet browser. They utilized an intrusion detection system (IDS) in their examining framework, which detects vulnerable JavaScript actions and takes suitable control measures that prevents violation in contradiction of the browser's security.

In general, behind the recommended strategy, the most thought is to ascertain occasions in which a JavaScript script is executed and the browser assets are utilized.

S. Gupta et al. [11] investigated performance concerns with existing cross-site scripting filters and suggested the XSS immune architecture as a result. The recommended approach is built on string assessment and JavaScript context enhancement. His system is built into the browser and associates the scripts existing within the HTTP request and reaction to identifying deceitful JavaScript code. Gupta [11] proposed a system that is able to decide the setting of distinguished XSS vulnerabilities before sterilizing them to decrease their effect on on-line web applications. Furthermore, the system can distinguish halfway script infusions, which can present destructive constraint values into the departing JavaScript.

M. Alenezi et al. [12 looked at a number of open-source projects to discover several types of web weaknesses including their origins. The researchers discovered various weaknesses including SQL injection and cross-site scripting in its different forms of vulnerabilities, which they attributed to programmer error, poor programming methods, and maintenance and enhancement operations. The author further argued that while open source projects are extensively employed by businesses to save money, they also expose them to security threats. They also gave two recommendations for dealing with security weaknesses. A system should assess the size, nature and types of attacks before choosing an open-source platform. Companies ought to create protected frameworks for development and encourage their employees to use the best secure programming techniques to eliminate code loopholes. The framework should be adaptable enough to choose prior projects and fix their flaws. The framework must be dynamic and quickly updatable to meet the most recent threats and vulnerabilities.

To avoid security vulnerabilities, S. Cho et al. [13] proposed a technique that is used at runtime a code is validated and security is enforced. The proposed method is used with applications written in Java. The method ensures input value verification at runtime using static bytecode composition.

I Medeiros [14] suggested a methodology that combines two approaches that can spot security flaws in web applications with the fewest possible incorrect detections. The researcher's technique merged taint analysis with a methodology that guesses the presence of false positives in the discovered flaws. They offered a method that combined taint analysis and machine learning.

D. G. Kumar and M. Chatterjee [15] proposed a construct for detecting SQL injection attacks. They applied the concept of information theory to detect SQL injections. The suggested framework is functional equally on the client and server sides. The main aim is to develop a client-side filter algorithm that verifies the attribute and size of a given variable, as well as issuing keyword cautions and after a preliminary evaluation, insert sensitive characters on the Client's side. The proposed method is divided into two phases on the server-side namely training and detection. The query's difficulty is estimated statically during the training phase and dynamically when submitted, and this is referred to as the query's entropy. Whenever the query's structure is changed as a result of attack inputs, the query's entropy will change automatically, resulting in a change in the related MAC value. SQL injection is indicated by a change in MAC values.

B. *Current Technologies*.

The researcher will study the following open source web scanners tools to identify strengths and weaknesses :

- Wapiti
- Websecurify
- Arachni
- W3af
- Vega

1 *Wapiti*

Wapiti allows its users to review security of their software applications running on a web server. A black-box methodology is used in this case where it does not have anything to do with the code, rather it peruse through the internet pages and detects forms where it can embed or infuse information.

2 *Websecurify*

Websecurify features a graphical client interface making it exceptionally simple to utilize. A test, once started, is performed sequentially. There are exceptionally few choices for customization. It is, in fact an effective instrument that affords programmed or manual checking methodologies. Once a weakness is found, it is displayed at the conclusion of the filtering handle. The report displays all the vulnerabilities identified together with the prescribed solutions.

3 *Arachni*

Arachni is known to give cognizance of the energetic nature of the applications and applies the required methodologies to identify escape clauses and loop holes in such requests. This tool is designed to function properly inside a web browser and hence it can identify client-side code and can utilize cutting-edge web advancement innovations.

4 *W3AF (Web application attack and audit framework)*

W3AF creates a system to help clients distinguish all sorts of web vulnerabilities. Results of the scan are shown on an HTML record when complete.

This open-source instrument is broadly utilized to check websites, primarily since it underpins HTTP and HTTPS, conjointly intuitiveness gives discoveries. W3AF is separated into two fundamental parts, the main, and the plug-ins. The core coordinates the methods and gives highlights that are devoured by the plug-ins, which discover the vulnerabilities and exploit them. The plug-ins are coupled and share data employing an information base.

5 Vega

When matched to other scanners used in this study, Vega produced one of the best results, owing to the fact that the vulnerabilities discovered are divided into four distinct categories: High, medium, low and data. This categorization is exceptionally valuable and gives a direction to the client on the exposures that must to be given precedence when fixing shortcomings.

C Algorithms Used by Existing Tools.

Shortcomings exist in the algorithms that are currently being used to detect vulnerabilities because available tools cannot detect all weaknesses in web applications. At times the tools report incorrect results like reporting vulnerabilities that do not exist. They tools are not 100% accurate hence have limitations. There is no safety and security guarantee of the web application even after scanning for vulnerabilities, [17]. Reporting vulnerabilities that do not exist is called "false positives", [16].

The researcher has concluded that there is no universal tool when it comes to web security. Checking a code before publication help in determining and closing out on loopholes within web applications.

D Current algorithms weaknesses.

Some of the tools in use currently are not designed to check certain types of vulnerabilities. Many researchers have done a lot of work towards web application security. Bau et al (2010), [18] proved that web security need to be improved by testing eight web vulnerability scanners (WVSs). Khoury (2011), [19] also did some analysis on stored SQLI by checking on three black box WVSs. These researchers' experiments proved that when these mechanized tools for scanning are initiated to exploit the type of vulnerability, stored SQLI might not be recognized indeed.

There are techniques that use black box testing to reduce false positives as discussed by Myers (2011), [21]. Fonseca et al (2014)[20] also tested for both SQLI and XSS detection performance of 3WVSs using preset tools composed with other fault injection methodologies.

III. PROBLEM DEFINITION

The popularity of online-based applications has skyrocketed as web technologies progress and users transition to online(web-based) solutions from conventional desktop applications. In the midst of highly skilled officials who program the architecture of the internet, there are amateur developers with almost zero knowledge on security who create web applications. Subsequently, there is exposure of web applications to a diverse mesh of security flaws that allow attackers to obtain unauthorized access to sensitive data.

There has been a rise in cybercrimes in Zimbabwe since the introduction of industry 5.0 which encourages companies to go digital. A number of financial organizations' online banking web applications have been attacked. Banks are losing millions of dollars as hackers are gaining access to their web systems through manipulating vulnerabilities in these systems.

IV. METHODOLOGY / APPROACH

A Proposed Hybrid Algorithm

The present algorithms have flaws, which is why they take a long time to scan online applications and the results aren't always accurate. The proposed solution is a hybrid algorithm whose objectives are to increase detection precision and increase proficiency of web scanners.

To improve application scanners, a black box technique was used. Divide and Conquer technique was used in the construction of this algorithm. The program was designed to crawl all web forms in the given web application URL and separately search for exposures.

There are five steps to the proposed method. This phase, also referred as assessment or trawling, focuses on collecting data more about web application. The further information gathered at this step, so the comprehensive the scanning procedure will just be. Following the completion of the first step, the Screening procedure begins, that entails identifying the web application's flaws. After the weaknesses are found, the following task is to understand them, and finally, an output is generated there at end of the transaction. The following are the steps the hybrid method goes through:

1. Inspection
2. Scanning
3. Vulnerabilities
4. Analysis
5. Report

B Proposed Hybrid Algorithm System Architecture.

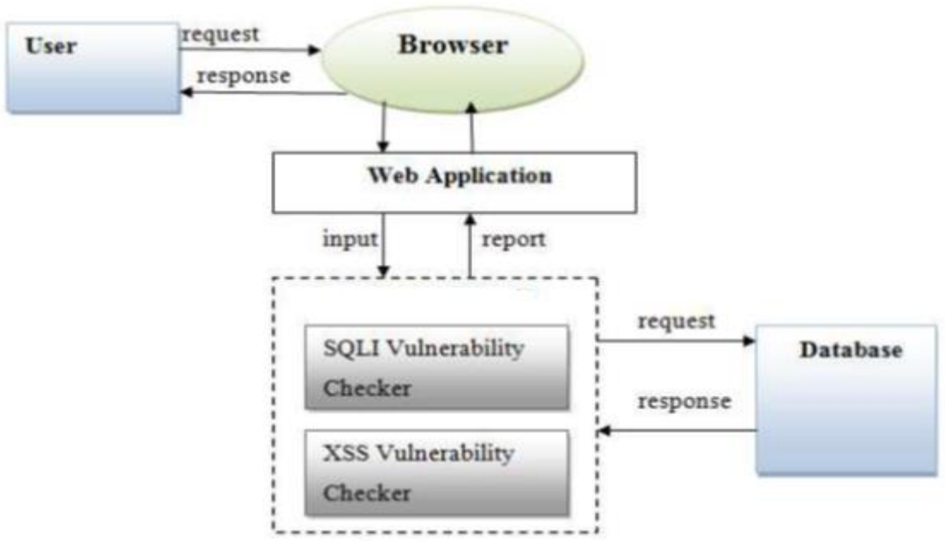

Fig. 1 Hybrid algorithm system architecture

V. IMPLEMENTATION OF THE PROPOSED TOOL

To identify SQL injection and cross-site scripting threats, the suggested solution will use a hybrid approach. In recent findings of interest is the hybrid approach where white-box and black-box testing techniques are joined to produce a model that minimizes false alarm rate from white-box testing by including black-box testing..

A Components of the hybrid algorithm.

i. **Crawler** - This part of the program establishes a connection to the website and browses all the web forms of the web pages gathering evidence about the application.

ii. **Fuzzing component** – This component of a web scanner handles responses and identifies vulnerabilities.

iii. **Report generator**. – This component organizes the scan findings and displays them in an appropriate format.

iv. **Report Saver**

This allows the tool to save the results in a text file for analyses.

B Testing the Algorithm.

The simulation was carried out with the help of a Python application. The simulation was put to the test against three web apps, with data collected on detection accuracy, dependability, consistency, and the time it took to scan each one. The simulation's results were compared to those of other open source web scanners available online after the testing procedure.

Testing Web Applications URL

A. http://testphp.vulnweb.com/artists.php?artist=1
B. https://xss-game.appspot.com/level1/frame
C. http://www.webscantest.com/crosstraining/aboutyou.php

SQL INJECTION TESTING

Different attack scenarios and a thorough examination of web application's SQL input variables, query structure, and entry points were tested. Test data was fine-tuned depending on multiple entry points, and results recorded. The tests done revealed that SQL vulnerability is detected on the URL itself not the forms on the webpages.

XSS TESTING

All XSS attacks impact websites in some way through consumer user input. Malicious software might also come from a simplified form posted by people or a more extremely complicated source like a JSON script, an XML web API, or perhaps an exploited cookie. The web design company ought to be cognizant that perhaps the information is transmitted from a secondary source and therefore should not be authorized because it could expose a security flaw.

The results here show that the XSS was detected in one form and the report of the vulnerability is summarized using the above parameters. (action which is not shown here making the webpage vulnerable to hackers, inputs (name, type and value (in this case the value is capturing the cookie on a webpage which makes the web app prone to XSS), the http method which is GET.

VI. RESULTS AND DISCUSSION

A, B and C are web applications with vulnerabilities mentioned above. Below are the statistics of the time taken to scan each application. The graph and table below the comparison of Vega, Arachni, and the Hybrid tool

Time taken in Sec	Vega sql	Vega XSS	Arachni sql	Arachni xss	WebAppSys sql	WebAppSys xss
A	19	36.00	16	28	12.36	27.36
B	15	22.00	19	20	10	20.49
C	17	32.00	13	23	13	22

Table:1 Results of tools used

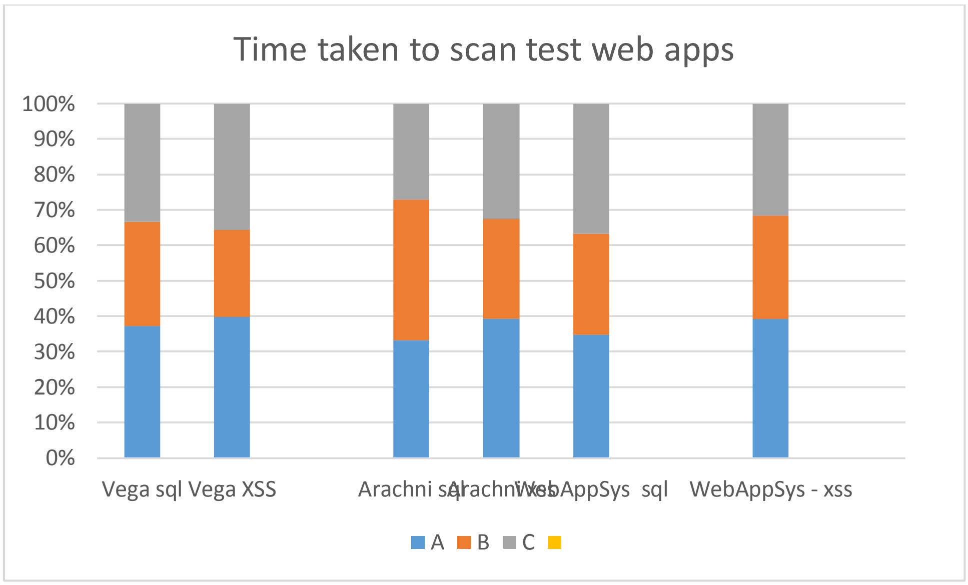

Fig. 2 Time taken to scan web apps

The proposed cross breed program is comprehensive when it comes to the implementation of its discovery component against applications that run on a web server's exposures. The suggested cross breed algorithm distinguishes extra imperfections. In any case, since the proposed crossover strategy did not filter all existing vulnerabilities, it is essential to extend the algorithm's slithering component to guarantee that "profound" slithering is done. Besides, the discoveries uncover that the proposed program has to be optimized in order to total the checking in a brief sum of time. More investigations are required to create an advanced algorithm competent of identifying other vulnerabilities.

VII. CONCLUSION

The open source tools can find exposures in the test cases that are run. Nevertheless, not any of the techniques are capable of detecting all of the flaws. McQuade arrived at the same conclusion (2014). The researcher concluded that in the world of information technology, there is no "silver bullet", and that any other black box vulnerability tools would be unable to detect all of the online vulnerabilities revealed by WAVSEP for comparison reasons. The researcher's hybrid tool is better than vega, one of the existing scanners on the market in terms of the time it takes to detect vulnerabilities and also even on detection accuracy.

VIII. FUTURE SCOPE

Because the hybrid method failed to detect all vulnerabilities, the fuzzing component of the algorithm must apply more advanced reasoning to improve the findings. The fuzzing component is in charge of triggering the required inputs in order to identify if they are any serious threats. The researcher's tool scans a web application in a shorter amount of time. The overall scanning processes of the hybrid algorithm must be improved in order to reduce scanning time without reducing detection accuracy. Standard security methods should be revised to include all viable security alternatives, rather than constraining developers to employ secure coding standards. A technique to detect new web application outbreaks with zero-day vulnerabilities is required.

ACKNOWLEDGEMENT

Without the amazing assistance and guidance of my supervisor, Mr. Mukosera, this research paper and the research that went into it would not have been feasible. Thank you so much to everyone who helped me with this evaluation.

REFERENCES

[1]. M. Parvez, P. Zavarsky and N. Khoury, "Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities," 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, 2015, pp. 186-191. doi: 10.1109/ICITST.2015.7412085
[2]. Awoleye, O. M., Ojuloge, B., & Ilori, M. O. (2014). Web application vulnerability assessment and policy direction towards a secure smart government. Government Information Quarterly, 31, S118-S125.
[3]. Pop, D. P., & Altar, A. (2014). Designing an MVC model for rapid web application development. Procedia Engineering, 69, 1172-1179.
[4]. Deepa, G., & Thilagam, P. S. (2016). Securing web applications from injection and logic vulnerabilities: Approaches and challenges. Information and Software Technology, 74, 160-180.
[5]. Asra Kalim, C K Jha, Deepak Singh Tomar, Divya Rishi Sahu, "A Framework for Web Application Vulnerability Detection, " International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-3, February, 2020.
[6]. OWASP, Top. "Top 10–2021 The Ten Most Critical Web Application Security Risks (2021). Accessed 13 December 2021 from https://owasp.org/www-project-top-ten/
[7]. Avinash Kumar Singh and Sangita Roy,"A network based vulnerability scanner for detecting SQLI attacks in web applications" in 1st International Conference of Recent Advances in Information Technology (RAIT), IEEE, 2012.
[8]. Asra Kalim, C K Jha, Deepak Singh Tomar, Divya Rishi Sahu, "A Framework for Web Application Vulnerability Detection" in International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-3, February, 2020. *Published By: Blue Eyes Intelligence Engineering & Sciences Publication*
[9]. Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo, "Securing web application code by static analysis and runtime protection," presented at the Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, 2004.
[10].O. Hallaraker and G. Vigna, "Detecting malicious JavaScript code in Mozilla," in *Engineering of Complex Computer Systems, 2005. ICECCS 2005. Proceedings. 10th IEEE International Conference on*, 2005, pp. 85-94.
[11].S. Gupta and B. B. Gupta, "XSS-immune: a Google chrome extension-based XSS defensive framework for contemporary platforms of web applications," *Security and Communication Networks,* vol. 9, pp. 3966-3986, 2016.
[12].M. Alenezi and Y. Javed, "Open source web application security: A static analysis approach," in *Proceedings - 2016 International Conference on Engineering and MIS, ICEMIS 2016*, 2016.
[13].S. Cho, G. Kim, S. J. Cho, J. Choi, M. Park, and S. Han, "Runtime input validation for Java web applications using static bytecode instrumentation," in *Proceedings of the 2016 Research in Adaptive and Convergent Systems, RACS 2016*, 2016, pp. 148-152.

[14].N. F. N. Ibéria Medeiros, Miguel Correia, "Automatic Detection and Correction of Web Application Vulnerabilities using Data Mining to Predict False Positives," presented at the Proceedings of the 23rd international conference on World wide web, Seoul, Korea, 2014.

[15].D. Kumar and M. Chatterjee, "MAC based solution for SQL injection," Journal of Computer Virology and Hacking Techniques, vol. 11, pp. 1-7, 2015/02/01 2015.

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

[17]. 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 2021]

[18]. Jason Bau, Elie Bursztein, Divij Gupta, John Mitchell," State of the Art: Automated Black-BoxWeb Application Vulnerability Testing" in IEEE Symposium on Security and Privacy, 2010.

[19]. 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.

[20]. 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.

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