ISSN 2278-3091
Volume 8, No.1.6, 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse0981.62019.pdf
https://doi.org/10.30534/ijatcse/2019/0981.62019

The Preparation of Cross-site Scripting in Automated Web Application Vulnerability Assessment: The Quantitative Analysis

Lim Kah Seng[1,2], Norafida Ithnin[2], Syed Zainudeen Mohd Shaid[2]
[1]OK Blockchain Centre Sdn Bhd, Johor, Malaysia
[2]School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia (UTM), Johor, Malaysia

ABSTRACT

Nowadays, practitioners have automated web application vulnerability assessment to speed up the testing life-cycle. Although this area of research had been widely studied worldwide for decades, however, existing studies show present state-of-the-art of automated web application vulnerability assessment still suffer from limitations of false alarms, which including both false positive and false negative. Therefore, this paper extends present research works by quantitatively analysing the web application security scanners' quality. The objective is to investigate present state-of-the-art performance in cross-site scripting detection for witnessing the decades of evolution. This paper achieves desired goal using the experimental research method, which the paper had quantitatively analysed six web application security scanner's performance for clarifying these scanners' capability in detecting the cross-site scripting. The experiment result shows present state-of-the-art still suffer from limitations of false positive, false negative and redundant test results.

Key words : Automated web application, Cross-site Scripting, False positive, False negative, Quantitative analysis.

1. INTRODUCTION

In this Age information, the web application is ubiquitous that web application has a crucial role in information sharing as well as communicating. Presently, the web application is widely used in economic, education, healthcare, politic, entertainment, and many more to spread information or get people connected. The web application is a unique and platform independent client-server application, which web application not only executable on a computer, mobile device, tablet of various operating systems, through the web browser, it is also accessible by anyone in 24/7, as long as there is an Internet connection. Moreover, the modern web application always possesses confidential data. Therefore, intruders always have their eyes on the vulnerable web application. According to a report of [1], the vulnerable web application is always a gate- way for intruders to gain access to confidential data. Unfortunately, awareness to strengthen web application security is rather low among the public. Henceforth, statistical reports of [2] and [3] show modern web application always possessing at least a security loophole and are vulnerable to vulnerabilities like SQL injection and cross-site scripting.

Existing countermeasure to the related problem has included educating and training developers to produce secure and high-quality web application, as well as assessing the web application security during and after the development phase [4-5]. Presently, there are software static analysis techniques like code debugging and code review for assessing web application security during the development phase. At the same time, there are also dynamic analysis testing techniques such as penetration testing to assess the fully developed web application security. However, manual testing is time consuming, tedious and error-prone. Consequently, practitioners have automating related testing process, leverage the computer's computation power. The introduction of automated web application vulnerability assessment not only succeeded in shortening the testing life-cycle but also allows tester performing the test in parallel.

A well-known invention is the web application security scanner, a tool that automates the process of web application vulnerability assessment. Unfortunately, according to experimental results of [6] and [7], current state-of-the-art of automated web application vulnerability assessment suffers from limitations of low test coverage, and are tend to produce the false alarms, which including both the false positive and false negative. In addition to that, these research works also show present state-of-the-art does not perform well in detecting security loopholes other than simple injection-based vulnerabilities like reflected cross-site scripting and reflected SQL injection. Consequently, a quantitative study was conducted in this paper to clarify present automated web application vulnerability assessment's quality for investigating advancement of the state-of- the-art of automated web application vulnerability assessment.

1.1 Related Works

The study of automated web application vulnerability assessment had existed for decades. Presently, practitioners had quantitatively quantified automated web application

vulnerability assessment performance through studying web application security scanners' performances, such as Acunetix Web Vulnerability Scanner (WVS), HailStorm, WebInSpect, AppScan, McAfee SECURE, Qualys Guard PCI, NeXPose, in detecting vulnerabilities like cross-site scripting, SQL injection, arbitrary file upload, remote file inclusion, OS command injection, code injection, session fixation, session prediction, cross-site request forgery, SSL misconfiguration, insecure HTTP method, insecure temporary file, path traversal, source code disclosure and error message disclosure. These experimentations' experimental results showed present state-of-the-art are good at detecting the SQL injection only. The detection rate of other vulnerabilities is low, which the tested scanners showed the vulnerability detection rate of less than 60% [6]. In the meanwhile, [8] had quantitatively measured present web application security scanners' quality with parameters of accuracy, time, and costs. Related experimental outcomes showed the web application security scanners reported high false positive and false negative, and this limitation had caused automated web application vulnerability assessment requiring more testing time than the manual testing. Besides this, [9] had quantitatively compared performances of automated static analysis tools and web application security scanners. The experimental outcomes showed false positive is common in automated static analysis tools, while web application security scanners are susceptible to the false negatives. Then, [10] and [11] studies showed present web application security scanners are weak at detecting stored SQL injection and stored cross-site scripting, which the scanners have failed to reach hidden web pages that had the attack string executed. On the other hand, [12] had evaluated Wapiti, Skipfish, as well as Arachni performance. Related experimental results showed the scanners had succeeded in detecting all the SQL injection but with the cross-site scripting detection rate of 87% only. Besides this, [13] had classified existing defensive mechanisms of those for securing web application from cross-site scripting, while [14] had developed Fire Range for benchmarking web application security scanners' quality in detecting the cross-site scripting. Overall, the experimental results showed web application security scanners contain limitations of the false positive and false negative. In addition to that, these research works also showed present web application security scanners only performed well in detecting the SQL injection but not the other web vulnerabilities. Consequently, this paper extended existing research works by quantitatively analysing current web application security scanners' state-of-the-art in detecting cross-site scripting.

1.2 Web application security scanner

Web application security scanner is a computer program invented to aid test engineer in assessing a web application security automatically. Web application security scanner simulates red team's actions, compromises web application confidentiality, integrity, and availability for vulnerability detection [15][16]. Presently, web application security scanner is designed to detect the well-known vulnerabilities, as reported in [17] and [18], with both SQL injection and cross-site scripting was the one that receives the most attention. However, research works of [4], [8], and [19] show present state-of-the-art merely excels in detecting the SQL injection but not the cross- site scripting. Consequently, this paper has defined the study scope to cover the cross-site scripting only

1.3 Cross-site Scripting

Cross-site scripting is a security loophole that allows execution of invalidated or malicious client-side scripts. The history of cross-site scripting is closely related to the invention of client-side technology named JavaScript by Netscape. Consequently, this vulnerability had long existed since the 1990s. Related client-side technology is invented to enhance web application responsive, interactivity, and presentation. Unfortunately, this client-side technology possesses an extra feature that allows cross-site browsing without the legitimate user consent. As a result, the attacker leverages this security loophole to achieve cross-site browsing for session stealing, legitimate user masquerading, credential information stealing, and many more. The cross-site scripting attack involves crafting and injecting the unsanitized web application with the malicious script to lure the legitimate user into executing the planted malicious scripts. Conventionally, practitioners classify cross-site scripting into three main categories based on its exploitation technique namely reflected cross-site scripting, persistent cross-site scripting, and DOM-based cross-site scripting [20][21]. Then, [22] and [23] successfully discovered another two cross-site scripting exploitation techniques called UTF-7 cross-site scripting and mutation cross-site scripting (mXSS). Figure 1 shows the taxonomy of cross-site scripting with both mutation cross-site scripting and UTF-7 cross-site scripting included.

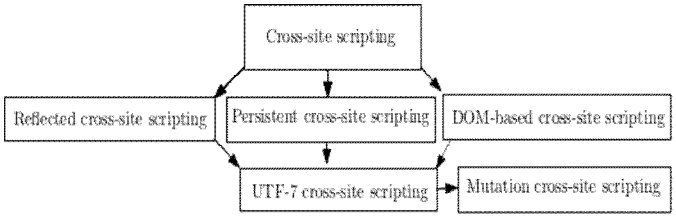

Figure 1: Cross-site scripting taxonomy

The reflected, persistent, and DOM-based cross-site scripting are the conventional cross-site scripting exploitation techniques well known by the public. The reflected cross-site scripting always involves direct execution of malicious scripts that planted on the malicious page. Persistent cross-site scripting, on the other hand, gets its name due to a fact that the malicious script is permanently writing to target web application, especially the database. In the meanwhile, DOM-based cross-site scripting is a client-side scripting vulnerability, which this cross-site

scripting attack directly modifies vulnerable web page's DOM document without sending the malicious script to the web server [21]. Then, [22] and [23] discover the mutation cross-site scripting and UTF7 cross-site scripting, which are the subclass of conventional reflected cross-site scripting, persistent cross-site scripting, and DOM-based cross-site scripting. Mutation cross-site scripting leverages browser default feature to convert the innocent string of text into malicious and executable client-side scripts. Similarly, mutation cross-site leverages default nature of modern web browser's to encode innocent text into the executable malicious script. Details regarding both mutation cross-site scripting and UTF7 cross-site scripting are reachable in [22] and [23].

1.4 Contribution and Organizations

Even though the experimental outcomes of [6], [8], [12] and [19] shows present state-of-the-art of automated web application vulnerability assessment has excelled in detecting the SQL injection, unfortunately, current state-of-the-art is not capable of detecting the cross-site scripting vulnerability. Therefore, this paper had conducted experimental research to quantitatively analyse current web application security scanners' quality for investigating whether improvements done on the state-of-the-art of automated web application vulnerability assessment did enhance the state-of-the-art performance in detecting the cross-site scripting. In summary, the paper had achieved the following contributions:

- We deliver the architecture and state-of-the-art of automated web application vulnerability assessment.
- We deliver the strengths and limitations of automated web application vulnerability assessment.
- We deliver web application security scanner's capability in detecting the cross-site scripting.

Overall, the research paper is divisible into five main sections. The remaining Section 2 presents the experimental research methodology. Section 3 presents the experimental results. Section 4 discusses the research findings. Lastly, Section 5 concludes this research paper.

2. THE EXPERIMENT METHOLOGICAL APPROACHES

The paper had performed the experiment to quantitatively analyse web application security scanner's performance in detecting the cross-site scripting. The performance of chosen web application security scanners was analysed using the experimentation framework of [9]. This experimental framework quantitatively measures the web application security scanner's performance with steps of preparation, execution, verification, and analysis. Figure 2 illustrates the [9] experimentation framework.

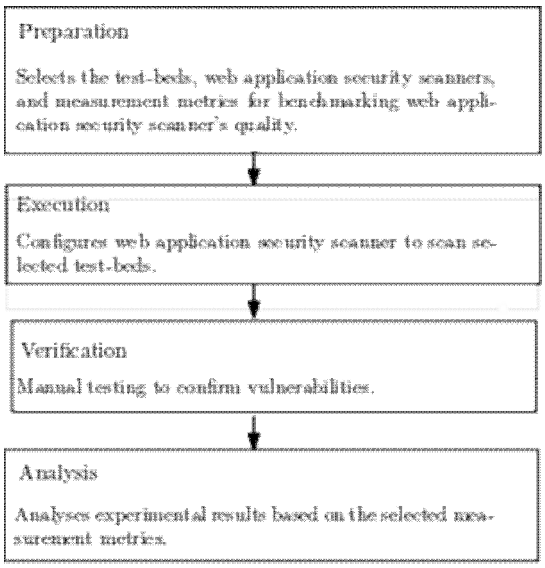

Figure 2: The methodology of [9]'s experimental framework.

The experimentation framework analysed web application security scanners' performance by having the web application security scanners scan selected test-beds, which are the vulnerable web applications. To ensure the experimentation was conducted in a sustainable and secure environment, we decided to use the virtual web application penetration testing lab. This virtual web application penetration testing lab was built upon guidelines provided by [24]. Figure 3 shows the virtual web application penetration testing lab's set-up.

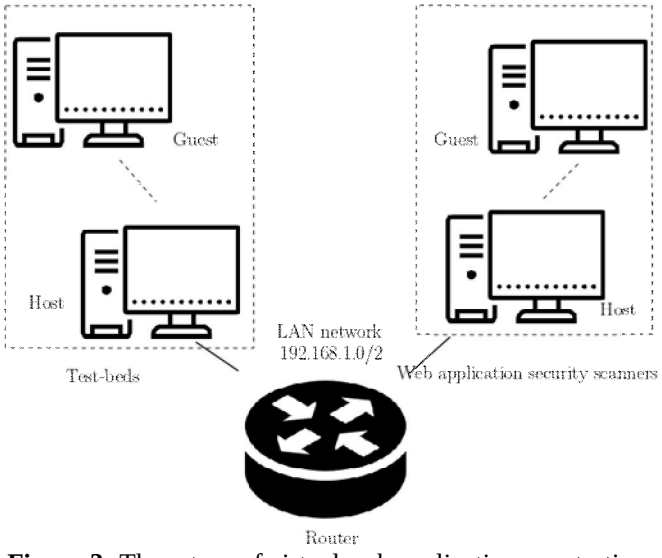

Figure 3: The set-up of virtual web application penetration testing lab.

As depicted in Figure 3, the virtual web application penetration testing lab was built using two PCs with specifications of Intel I7 processor and 8GB memory. These two machines were connected through a router for producing the isolated local area network, to prevent the

web application security scanners from accidentally scanning the World Wide Web (WWW). Besides this, in each machine, a guest machine was created using the virtualization technology. The virtualization was used to enable the machines to support those web application security scanners that executable on Linux platform only. In the meanwhile, the container technology of docker was used to host the test-beds.

2.1 Preparation

The preparation phase involves the selection of desired test-beds and web application security scanners, as well as the suitable measurement metrics to measure web application security scanners' quality. Overall, a total number of eleven test-beds and six web application security scanners were chosen to be evaluated in this quantitative analysis using the measurement metrics of false positives, false negatives, redundant tests, and true negative, as previously practised by [9]. These test-beds and web application security scanners were chosen based on criteria of [25] that they are easy to use and install; they are ubiquitous; they are free or not too expensive; they support cross-site scripting vulnerability, and they are well-documented. A simple reason that test-beds, which are the vulnerable web application, were used to benchmark the web application security scanner's quality was to prevent the committing of cybercrime as in the web application security scanner might harms a web application security through the active scanning, which involves writing of attack payloads into the web application attack vectors and database. Table 1 presents the selected test-beds and web application security scanners.

Table 1:. The Selected Test-beds and Web Application Security Scanners

Object	Name	Technologies
Test-beds	WackoPicko	PHP
Test-beds	Peruggia	PHP
Test-beds	bWAPP	PHP
Test-beds	Bodgeit	JAVA
Test-beds	JavaVulnerableLab	JAVA
Test-beds	Mutillidae II	PHP
Test-beds	Acuart	PHP
Test-beds	SecurityTweet	Python
Test-beds	AcuForum	ASP
Test-beds	AcuBlog	ASP.NET
Web Application Security Scanner	W3af	JAVA
Web Application Security Scanner	Skipfish	C
Web Application Security Scanner	ZAP Proxy	JAVA
Web Application Security Scanner	WebScarab	JAVA
Web Application Security Scanner	Paros Proxy	JAVA
Web Application Security Scanner	Wapiti	Python

2.2 Execution

The execution phase involves configuring and installing the chosen web application security scanners and test-beds. Afterwards, having the web application security scanners executed to scan the selected test-beds. During the experimentation, the configurations and settings of selected web application security scanners and test-beds were configured following the guideline written in the documentation.

2.3. Verification

The verification phase involves the inspection of test results yielded by the selected web application security scanner. In this activity, the test reports validity was clarified manually using the manual testing.

2.4. Analysis

The analysis phase quantifies web application security scanners' quality with measurement metrics of false positives, false negative, true negatives, and duplicate results.

3. THE QUANTITATIVE ANALYSIS

Experimentation had been conducted in this paper to quantitatively quantify present web application security scanners' performance in detecting the cross-site scripting. Figure 4 presents quantitative analysis outcomes.

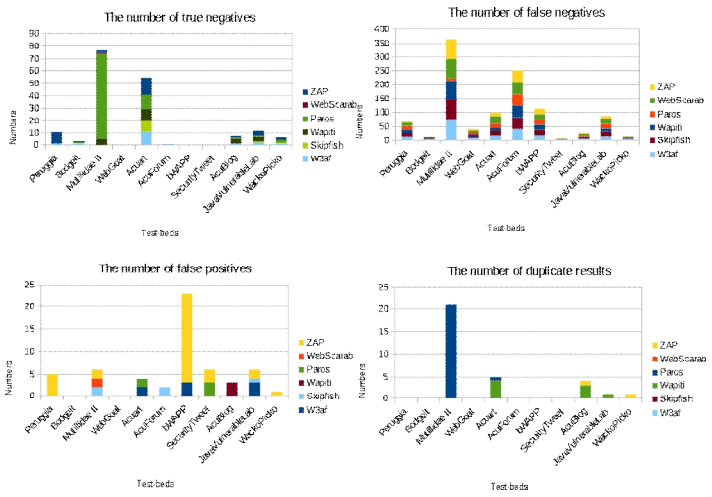

Figure 3: The quantitative analysis outcomes

3.1. The Number of True Negatives

The measurement metric of the number of true negatives illustrates the benign cross-site scripting vulnerabilities that a web application security scanner has successfully detected [27]. According to the experimental outcome of Figure 4, current web application security scanners only manage to detect cross-site scripting in some test-beds, which have

simple business logic and without the authentication scheme. Henceforth, the selected web application security scanner was able to efficiently detect cross-site scripting vulnerabilities in Acuart but not the others test-beds. This experimental outcome was observed due to a fact that these test-beds were custom built, hence, whenever the web application security scanners scanned these test-beds in the testing environment of black-box, the web application security scanners failed to precisely locate the cross-site scripting. Consequently, the number of benign cross-site scripting vulnerabilities reported by the chosen web application security scanner was generally low in number.

3.2. The Number of False Negatives

The measurement metric of false negative illustrates fake cross-site scripting vulnerabilities that the web application security scanners were reporting during the quantitative analysis [28]. Figure 4 shows all the selected web application security scanners yielded at least a false negative. The outcome of the analysis showed the authentication mechanism in the test-beds had prevented the chosen web application security scanners from reaching the hidden web contents, causing the cross-site scripting security loopholes embedded in hidden web contents not detected by the web application security scanners. Therefore, false negatives were produced by chosen web application security scanners, when scanning test-beds of Mutillidae II, bWAPP, JavaVulnera- bleLab and WebGoat. The chosen web application security scanners were just skipped these web contents as it had never existed and ended the testing session with an incomplete test. Thus, we conclude that incomplete testing was the primary factor that contributes to the research problem of false negatives.

3.3. The Number of False Positives

The measurement metric of false positive illustrates fake cross-site scripting vulnerabilities yielded by the web application security scanners [27]. Figure 4 shows selected web application security scanners had produced false positive results while scanning the test-beds. The quantitative analysis showed current vulnerability detection techniques of signature-based vulnerability detecting mechanism and learning-based vulnerability detecting mechanism are too conservative for assessing the cross-site scripting vulnerabilities in test-beds. The web application security scanners were always mistakenly interpreted the sanitized attack payloads as the cross-site scripting vulnerabilities. Besides this, these web application security scanners also occasionally mistakenly assumed others security loopholes as the cross-site scripting vulnerabilities. However, the experimental results showed the number of false positives produced by these web application security scanners was rather low compared to the number of false negatives.

3.4 Duplicate Results

The duplicate cross-site scripting illustrates duplicated cross-site scripting that the web application security scanners had produced during the quantitative analysis. Figure 4 showed limitation in the present crawling mechanism had caused current web application security scanners yielded the duplicate cross-site scripting, which the weak crawling made the web application security scanners visits similar web contents for more than once. The phenomenon was particularly common when the web application security scanners assessed the test-beds that with a lot of redundant and self-referencing links, such as the test-beds of Mutillidae II, Acuart and AcuBlog.

4. DISCUSSION

The quantitative analysis shows present automated web application vulnerability assessment's state-of-the-art still suffers from limitations of false positive, false negative, and redundant test result, even after the decades of evolution. Besides this, this study reveals existing web application security scanners generally consist of three main components, which are web application reconnaissance component, vulnerability assessment component, as well as the vulnerability detection component.

The web application reconnaissance solutions generally comprised of a web crawling mechanism that responsible for reverse-engineer the target web application for discovery the data entry points and attack vectors. This web crawling mechanism generally is a combination of a web crawler, a proxy, and a form inputting mechanism. The web crawler brute forces and statically parses web application tree structure to reach web application contents. In the meanwhile, the proxy is applied to inspect the propagation of HTTP requests and HTTP responses that flow between client and server. On the other hand, the form inputting mechanism asks inputs from the tester to inputting the web forms for reaching the hidden web contents.

The vulnerability assessment component then reads the attack vectors to perform the web exploitation, penetrating the attack vector security for compromising the web application confidentiality, integrity or availability. This component generally contains a brute forcing mechanism that consists of an attack vector selector, an attack string injector, and an attack library. The attack vector selector selects the desired attack vector from a pool of attack vectors. Subsequently, the string injector brute forces the attack vector with fuzzing or fault injection technique while the attack library delivers the payloads to penetrating the attack vectors security.

After the vulnerability assessment, the vulnerability detection component takes place by inspecting web application responses, locating the anomalies, for vulnerability detecting using the conventional signature-based vulnerability detection mechanism or learning-based vulnerability detection mechanism. A vulnerability is deemed existed whenever the attack string has executed and violated the security rules. Then, each detected vulnerability is recorded and displayed in a test report. Unfortunately, the existing automated web application vulnerability assessment's state-of-the-art suffers from limitations of false positive, false negative and redundant test results

4.1. The Issue of False Positive and False Negative

The quantitative analysis showed weak web application reconnaissance and vulnerability detection solutions have caused present state-of-the-art produces both false positive and false negative during the automated web application vulnerability assessment. Current reconnaissance solution had found failed to explore hidden web contents and bypassing the web application authentication scheme, causing many web contents not tested during the vulnerability assessment session. In addition to that, the weak crawling mechanism also had severely affected current web application security scanner's reputation with the duplicate test results. On the other hand, the existing vulnerability detection solutions are too conservative that they were not capable of precisely detecting the cross-site scripting vulnerabilities in deployed test-beds, resulted in the generation of both false positives and false negatives.

5. CONCLUSION AND FUTURE WORK

The state-of-the-art of automated web application vulnerability assessment had gone through the decades of evolution, yet the experimental analysis still showed current web application security scanners contained limitations of false positives, false negative, and redundant results. Presently, the state-of-the-art still heavily relies on the conventional approaches of web reconnaissance, vulnerability assessment and vulnerability detection with slow pace improvements. The quantitative analysis showed the weak reconnaissance and vulnerability detection solutions are the primary factor the selected web application security scanners suffer from limitations of false positive, false negative and redundant test results. The future work will involve improving the present state-of-the-art limitations to reduce the false positive, false negative and redundant results.

ACKNOWLEDGEMENTS

We would like to deliver the appreciations to OK BlockChain Centre Sdn. Bhd. for funding this quantitative analysis.

REFERENCES

1. A. Top, "500 Global Sites (2018)," URL Httpwww Alexa Comtopsites, 2018.
2. 2019 edgescan vulnerability Stats report..
3. Webapplication vulnerabilities: statistics for 2018. [Online]. Available: https://www.ptsecurity.com/ww-en/analytics/web-application-vulnerabilities-statistics-2018/. [Accessed: 13-Mar-2019].
4. Juskaite, Loreta. "Analysis and Comparison of the National Diagnostic Paper-Based and Online Tests." International Journal of Advanced Trends in Computer Science and Engineering, vol. 8, no. 1, 2019, pp. 280–86, doi:10.30534/ijatcse/2019/4981.12019.
5. Lefebvre, Olivier. "Mergers of Operators and Regulation: A Game – Theoretic Approach." International Journal of Advanced Trends in Computer Science and Engineering, vol. 8, no. 1, 2019, pp. 73–78, doi:10.30534/ijatcse/2019/1481.12019.
6. J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, "State of the art: Automated black-box web application vulnerability testing," in Security and Privacy (SP), 2010 IEEE Symposium on, 2010, pp. 332–345.
7. F. van der Loo, "Comparison of penetration testing tools for web applications," PhD Thesis, Master's thesis, University of Radboud, Netherlands, 2011.
8. L. Suto, "Analyzing the accuracy and time costs of web application security scanners," San Franc. Febr., 2010.
9. N. Antunes and M. Vieira, "Benchmarking vulnerability detection tools for web services," in Web Services (ICWS), 2010 IEEE International Conference on, 2010, pp. 203–210. https://doi.org/10.1109/ICWS.2010.76
10. N. Khoury, P. Zavarsky, D. Lindskog, and R. Ruhl, "An analysis of black-box web application security scanners against stored SQL injection," in Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, 2011, pp. 1095–1101.
11. 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," in Internet Tech- nology and Secured Transactions (ICITST), 2015 10th International Conference for, 2015, pp. 186–191.
12. M. Alsaleh, N. Alomar, M. Alshreef, A. Alarifi, and A. Al-Salman, "Performance-Based Com- parative Assessment of Open Source Web Vulnerability Scanners," Secur. Commun. Netw., vol. 2017, 2017.
13. S. Gupta and B. B. Gupta, "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art," Int. J. Syst. Assur. Eng. Manag., vol. 8, no. 1, pp. 512–530, 2017.
14. E. Bazzoli, C. Criscione, F. Maggi, and S. Zanero, "XSS Peeker: A Systematic Analysis of Cross-site Scripting Vulnerability Scanners," ArXiv Prepr. ArXiv14104207, 2014.
15. P. Baral, "Web application scanners: a review of related articles [Essay]," IEEE Potentials, vol. 30, no. 2, pp. 10–14, 2011. https://doi.org/10.1109/MPOT.2010.939449
16. E. Fong and V. Okun, "Web application scanners: definitions and functions," in System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on, 2007, pp. 280b–280b.
17. Z. Djuric, "A black-box testing tool for detecting SQL injection vulnerabilities," in Informatics and

Applications (ICIA), 2013 Second International Conference on, 2013, pp. 216–221.
18. D. E. Simos, K. Kleine, L. S. G. Ghandehari, B. Garn, and Y. Lei, “A Combinatorial Approach to Analyzing Cross-Site Scripting (XSS) Vulnerabilities in Web Application Security Testing,” in IFIP International Conference on Testing Software and Systems, 2016, pp. 70–85.
19. L. Suto, “Analyzing the effectiveness and coverage of Web application security scanners,” San Franc. Oct., 2007.
20. K. Spett, “Cross-site scripting,” SPI Labs, vol. 1, pp. 1–20, 2005.
21. J. Grossman, XSS Attacks: Cross-site scripting exploits and defense. Syngress, 2007. https://doi.org/10.1016/B978-159749154-9/50005-6
22. Y. Hasegawa, “UTF-7 XSS cheat sheet,” Retrieved At, p. 2, 2005.
23. M. Heiderich, J. Schwenk, T. Frosch, J. Magazinius, and E. Z. Yang, “mxss attacks: Attacking well-secured web-applications by using innerhtml mutations,” in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013, pp. 777–788.
24. K. Cardwell, Building Virtual Pentesting Labs for Advanced Penetration Testing. Packt Pub- lishing Ltd, 2016.
25. A. Tetskyi, V. Kharchenko, and D. Uzun, “Neural networks based choice of tools for penetra- tion testing of web applications,” in 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), 2018, pp. 402–405. https://doi.org/10.1109/DESSERT.2018.8409167
26. “alexa top site - Google Scholar.” [Online]. Available: https://scholar.google.com/scholar. [Ac- cessed: 03-Oct-2018].
27. Y.-H. Tung, S.-S. Tseng, J.-F. Shih, and H.-L. Shan, “W-VST: A Testbed for Evaluating Web Vulnerability Scanner,” in Quality Software (QSIC), 2014 14th International Conference on, 2014, pp. 228–233.
28. Y.-H. Tung, S.-S. Tseng, J.-F. Shih, and H.-L. Shan, “A cost-effective approach to evaluat- ing security vulnerability scanner,” in Network Operations and Management Symposium (AP- NOMS), 2013 15th Asia-Pacific, 2013, pp. 1–3.