International Journal of Computer Applications (0975 – 8887)
Volume 174 – No. 18, February 2021

Performance Evaluation of Open Source Web Application Vulnerability Scanners based on OWASP Benchmark

Pious Akwasi Sarpong
S.D.A Coll. Of Educ
Asokore-Koforidua, Ghana

Lawrence Sakyi Larbi
Presby Coll. Of Educ
Akropong-Akuapem

Daniel Paa Korsah
Komenda Col.l of Educ
Komenda, Ghana

Issah Bala Abdulai
Kibi Presby Coll. of Educ
Kibi, Ghana

Richard Amankwah
Presby Coll. Of Educ
Akropong-Akuapem, Akropong, Ghana

Akwasi Amponsah
Mamp. Tech Coll. of Educ.
Asante Mampong, Ghana

ABSTRACT

The use of web application has become a critical component in our daily routine work due to its enormous benefits. Unfortunately, most of the web application deployed are not totally devoid of bugs which makes them vulnerable to attacks. Web application scanners are tools that detect security vulnerability in web application. Although there are several commercial and open-source web application vulnerability scanners proposed in literature, the performance of these scanners varies in relation to their detection capabilities. The aim of this paper is to assess and compare the vulnerability detection capabilities of five open-source web application vulnerability scanners (WAVS), namely, ZAP, Skipfish, Arachni, IronWASP and Vega by executing them against two vulnerable web applications, damn vulnerable web application (DVWA) and WebGoat. Furthermore, we evaluate the performance of the scanner results using the OWASP benchmark metric. The experimental results show that ZAP, Skipfish and Vega are very efficient for detecting the most common web vulnerabilities, such as Command Execution Cross-Site Scripting and SQL injection. The findings further show Skipfish obtained the highest Youden index of 0.7 and 0.6 in DVWA and WebGoat, which makes the scanner superior than all the studied tools. Based on our evaluation results, we make some valuable recommendations since software security is a very fast-growing domain.

General Terms

Software security, web security, software engineering

Keywords

Web vulnerability scanner, web application, damn vulnerable web application, open-source scanners

1. INTRODUCTION

The use of web application has become inevitable in our daily life because it is widely applied in diverse domains such as banking, transportation, manufacturing, business and education. Due to a geometrical increase in the use of web application it has consequently resulted in an equally geometrical increase in its web attack. Software vulnerability according to Sagar et al. [1] are the weaknesses, flaws and errors in software systems. Command injection, buffer overflow, data manipulation, path manipulation, authentication, session hijacking, cookie misinterpretation are some of the categorize of security vulnerabilities [2]. These vulnerabilities normally cause data breaches and have serious security implications when exploited. For this purpose, a number of web application vulnerability scanners (WAVS) such as (W3af) [3] OWASP Zed Attack Proxy (OWASP ZAP) [4], Skipfish [5], Arachni, Vega, [6], Stalker and Iron WASP [7] emerged to address this phenomenon. Tung et al. [8] defined these WAVS as tools used to test and detect common security breaches in web application.

These tools are automated and provide an easy way of detecting security vulnerability in web applications in order develop mitigation strategies. Investigation, conducted in two annual vulnerability reports namely Open Web Application Security Project (OWASP) and the National Institute of Standard and Technology (NIST), shows that there are several web application vulnerability scanners with varied efficiency and detection capabilities. Similarly, previous studies such as the work of Antunes and Vieira [9], Makino and Kleve [10] and Parvez [11] affirmed the fact that there are several open source web vulnerability tools with diverse efficiency level and user friendliness [12]. But the question is which one of these open source web application vulnerability scanners are most suited for detecting a particular type of security vulnerability, have a high detection and a low false rate? In attempt to answer these questions there have been several comparative studies such as the ones proposed by Antunes [13], Fonseca et al. [14] and Suto [3] to investigate the performance of the tools. For example, Fonseca et al. [15] performed a comparative study by investigating the vulnerability detection capabilities of three web application vulnerability scanners. The authors assessed the effectiveness of the tools using evaluation metrics such as coverage and false positive. The finding shows that the three web scanners studied can effectively detect the two topmost web vulnerability, namely SQL injection and Cross Site Scripting (XSS).

In another study, Makino and Kleve [16] evaluated the detection effectiveness of two open source scanners, namely OWASP ZAP and Skipfish using Damn Vulnerable Web Application (DVWA) and Web Application Vulnerability Scanner Evaluation Project. The experimental result indicates that ZAP is superior to Skipfish.

Although there are several comparative studies on web application vulnerability scanners (i.e. Commercial and open source), we observe that the focus is mainly on the commercial scanners. Hence the aim of this study is to focus on the free/open source web application vulnerability scanners following a similar procedure by Makino and Kleve [16] to propose other alternative and easy to use web vulnerability detection scanner for vendors. As a result, we evaluate five open source web vulnerability scanner, namely OWASP Zed Attack Proxy (OWASP ZAP) [17], Skipfish [5], Arachni [6], Iron WASP [7] and Vega [6] by running them against two vulnerable web applications, damn vulnerable web application (DVWA) and WebGoat, which is a free and open test suite developed to evaluate the coverage, speed and accuracy of automated software vulnerability detection tools [18].

We further measure the performance of the scanners using OWASP benchmark metric which evaluate the performance and effectiveness of tools based on True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR) and False Negative Rate (FNR) [19], [10], [20]. Thus, this study makes the following contributions:

1. To evaluate the performance of five open source WAVS, namely ZAP, Skipfish, Arachni, IronWASP and Vega in identifying security vulnerability in web service environment using the DVWA and WebGoat.
2. To study the various limitations of the various open source scanners evaluated in this study.
3. Suggest possible measures that can be used to improve these open source web scanners.

The remaining sections of the paper are structured as follows: Section II presents a review of related works. Section III presents background of the study. Section IV discusses experimental setup. Section V details the methodology employed in the study. The results and discussion are presented in Section VI. Section VII conclude the study and provides lessons learned and recommendation for the future.

2. RELATED WORK

There have been a number of studies conducted to evaluate the performance of open-source web application vulnerability scanners to ascertain the most effective and recommend for vendors or serve as an alternative to the commercial scanners. For example, Sagar et al. [1] evaluated the detection capability of three open source web vulnerability scanner namely w3af, Skipfish and OWASP Zed Attack Proxy (ZAP). The authors assessed the performance of the web scanners using the Damn Vulnerable Web Application (DVWA). Parvez [11] conducted a comparative study by evaluating the effectiveness of three web application vulnerability scanners, namely Acunetix, AppScan and ZAP. The results show improved detection rate. Alsaleh et al. [21] assess the performance of four open source scanners, namely Arachni v0.4.3, Arachni v0.4.3, Wapiti, Skipfish. The result showed similar detection rate for the four scanners evaluated in the study. Similarly, Vieira et al. [4] evaluated four commercial vulnerability scanners, namely WebInspect, AppScan, WSDigger and Wsfuzzer to detect security flaws in web application.

The authors conducted an experiment using 300 well known web application. The findings show that the selected scanner generates a number of false positives between 35% and 40%. Although, there is a large number of studies on this domain, we observe that most of the evaluations are based on the commercial web application vulnerability scanners. Again, most of these studies are focused on only SQL injection and cross site scripting. Lastly, we observe that most studies do not examine and compare the performance scanners based on the Damn Vulnerable Web Application and WebGoat. Hence, the current study evaluates and assess the performance of five web scanners based on the aforementioned vulnerable web application and suggest possible recommendation for future research direction in this domain of stud. To the best of our knowledge our study is novel and we contribute new knowledge to the research domain

3. OVERVIEW OF WEB APPLICATION SCANNERS AND VULNERABILITIES

This section of the paper is divided into two sub-section. The first section presents brief overview of web application vulnerability scanners (WVS). The second section discusses web application vulnerabilities that is examined in this study.

3.1 Overview of Web Application Vulnerability Scanners (WAVS)

Web application vulnerability scanner (WAVS) examines an application by going through its web pages and performs penetration testing. Most WAVS consist of three main components: (1) a crawling component, an attacker component, and an analysis component [22]. The crawling component identifies all related and input pages of the web application, after the user enters the Uniform Resource Locator (URL) of the web application in the scanner. The attacker component breakdown discovered information from the various webpage for each of the input vector, vulnerability type and send content to the web server. The analysis component evaluates and interpret the response from the server, if a given attack was successful or not. Basically there are two main techniques used to test web application for available vulnerability [23]: White box testing: This involves analysis the source code of the web application either manually or using a code analysis tools. The black box testing on the hand executes the application in order to detect and locate security vulnerabilities. This techniques is normally referred to as penetration testing [24]. Ashcan [25], Web King [26], Web Inspect [27] Topsider [28] are some of the most widely applied commercial web application scanners: Other web application scanners can also be accessed in the following studies [29], [30], [31].

3.2 Overview of Web Application Vulnerabilities Tested

We used web application vulnerabilities listed by the National Vulnerability Database in partnership with the National Institute of Standards and Technology (NIST).

Currently, the database contains over 9900 security vulnerabilities in diverse software product. Several empirical studies, [32], [33], [34] have successfully applied/evaluated this vulnerabilities. The current study focuses on identifying the existence of these vulnerabilities in DVWA and WebGoat.

4. METHODOLOGY AND EXPERIMENT

In this section of the study, we present the methodology and the experimental setup.

4.1 Methodology

We conducted an investigative study to identify and review the most widely applied open source web vulnerability scanner based on predefine criteria offered by Web Application Security Consortium. We then scanned for vulnerabilities using the selected scanners randomly in WebGoat and DVWA by configuring our browser and the selected scanners to work with DVWA and WebGoat.

We further perform a detailed analysis of the result produced by the various scanners after the detection stage. Finally, we analyzed and compare the scanners performance using the OWASP benchmark metric to determine the tools precision, recall and Youden index to ascertain which scanner is most effective and superior in detecting security vulnerability in web applications.

4.2 Experimental setup

The experimental activity is divided into three steps: Pre-Experimental Activities, Experimental Activities and Post Experimental Activities.

4.3 Open Source Web Vulnerability Scanner

As stated earlier, we used five open source scanners, namely ZAP, Skipfish, Arachni, W3af, N. Stalker, IronWASP and Vega. ZAP [35] is an open source web vulnerability scanner with a user friendly interface used for penetration testing. It can be used by people with different capabilities in the field of software security.

Skipfish [36] is an active web application security reconnaissance tool. It prepares an interactive sitemap for the targeted site by carrying out a recursive crawl and dictionary-based probes. The resulting map is then annotated with the output from a number of active security checks. The final report generated by the tool is meant to serve as a foundation for professional web application security assessments. Arachni [37] is very effective and user-friendly web security vulnerability too written in Ruby. It is very fast in scanning and offers different user interface. Again, it provides a customized, command driven input and its output is in the form of HTML. Iron WASP (Iron Web application Advanced Security testing Platform) is a very powerful web application advanced security testing platform which comes in various external libraries such as IronPython, IronRuby, Json. NET etc. Vega an automated open-source web vulnerability scanner for detecting SQL and other vulnerability type.

The aforementioned scanners were selected for the study based on a comparison criteria proposed by the Web Application Security Consortium, “Web Application Security Scanner Evaluation”[38] and another study conducted by Suteva et al. [39] on the most popular open source vulnerability scanners. The scanners were run on a workstation with an Intel(R) Core (TM) i5-6500 CPU at 3.20GHz, 4 GB of RAM and Windows 7 Ultimate. All the scanners evaluated has a graphical user interface and run on Windows. Additionally, Skipfish, Arachni and W3AF are available on FreeBSD. Table 1 list the features of the five open source WAVS used in our study.

Table 1: Features of Scanners

Scanner	Company	Platform	Version	Operation
ZAP	OWASP	Java	2.7.0	Windows, Linux, OS X
Skipfish	Google	Java	2.10beta	Windows, Linux, OS X, FreeBSD
Arachni	Arachni	Java	1.5.1-0.5.12	Windows, Solaris, Linux, BSD, Unix
IronWASP	IronWASP	Java	0.9.8.6	Windows, Linux. OS X
Vega	Subgraph	Java	1.0	Windows, Linux, OS X

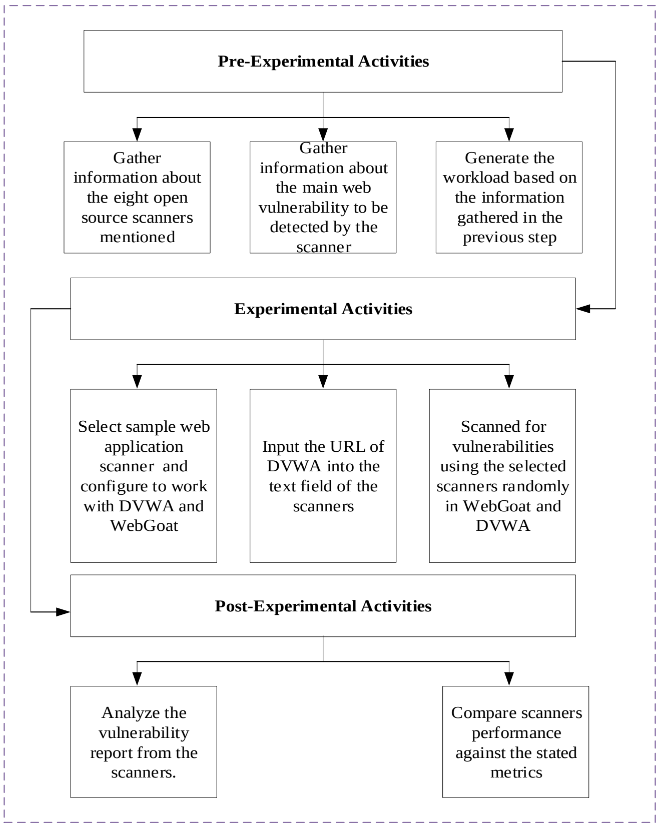
Pre-Experimental Activities
Gather information about the eight open source scanners mentioned
Gather information about the main web vulnerability to be detected by the scanner
Generate the workload based on the information gathered in the previous step
Experimental Activities
Select sample web application scanner and configure to work with DVWA and WebGoat
Input the URL of DVWA into the text field of the scanners
Scanned for vulnerabilities using the selected scanners randomly in WebGoat and DVWA
Post-Experimental Activities
Analyze the vulnerability report from the scanners.
Compare scanners performance against the stated metrics

Figure 1*:* Experimental Activities

4.4 Benchmark Vulnerable Web Application Selection

There are several benchmarks vulnerable web applications, namely open web application security project (OWASP) benchmark [40] ,web application vulnerability scanner evaluation project (WAVSEP) benchmark [41], web input vector extractor teaser (WIVET) [42], WebGoat benchmark [43] and the damn vulnerable web application (DVWA) benchmark [44]. WAVSEP, OWASP and WIVET have been applied mostly to evaluate commercial and open source scanners [45], [41], [46]. Conversely, DVWA and WebGoat benchmark have not been used much to evaluate many popular WAVS although it is developed by a well-known organization and is actively maintained. Hence, we used DVWA and WebGoat to evaluate the performance of the tools (i.e.to obtain the true positives, false positives and the false negatives.) since they are regarded as one of the best benchmark option for assessing the effectiveness of web application vulnerability scanners [19]. DVWA [44] has a friendly user interface that allows developer, teachers and students to explore and analyze web service security. It consists of vulnerabilities such as Command Execution, Cross Site Request Forgery, Insecure captcha, File inclusion, SQL injection, SQL injection (blind), Reflected Cross-site scripting (XSS), Stored Cross-site scripting (XSS). WebGoat [43] on the other hand is an open source OWASP application created to help developers and experts test their tools in detecting security vulnerability in web application.

The vulnerability types in WebGoat includes the following: Access Control Flaws, AJAX Security, Authentication Flaws, Buffer Overflows, Code Quality, Concurrency Cross-Site Scripting, Bypass Error Handling, Injection Flaws, Denial of Service, Insecure Communication, Insecure Configuration, Insecure Storage, Malicious Execution, Parameter Tempering, Session Management, Web Services.

4.5 Evaluation Metric

The evaluation of this study is conducted in two stages. In the first stage we obtained the benchmarking results (i.e. TPR, FPR, TNR and FNR) by first executing the scanners against the two vulnerable web applications, damn vulnerable web application (DVWA) and WebGoat. Secondly, evaluate each tool performance based on precision, recall and Youden index to be able to make inform conclusions for each of the scanner under study. The performance metrics are:

Precision is defined in [40] as the percentage of correctly detected bugs to the number of all detected bugs (i.e. number of bugs detected by the tool that are actually rear bugs). Eq. 1, shows the formula for this metric. A precision value of 100% represents a high detection accuracy of the exact bug.

$$\mathbf{Precision} = \frac{\mathbf{TP}}{\mathbf{TP + FP}} \qquad \text{(Eq.1)}$$

Recall [47] is the percentage of the correctly detected bugs to the number of known bugs (i.e. a number of bugs that were supposed to be detected by the tool but couldn't detect. Eq. 2 shows the formula for recall.

$$\mathbf{Recall} = \frac{\mathbf{TP}}{\mathbf{TP + FN}} \qquad \text{(Eq.2)}$$

The Youden index (Yi) [48] was proposed by W.J. Youden to evaluate the performance of analytical tests (diagnostic tests). The values for the index range from -1 to 1. For instance, if a tool is able to detect all bugs without any false positive present it obtains a Youden index of 1.

However, if the tool could not detect actual bugs but produced false positives then it obtains a Youden index of -1. A Youden index of 0 is invalid. Eq. 3 shows the formula for Youden index.

$$\mathbf{Yi} = \frac{\mathbf{TP}}{\mathbf{TP + FN}} + \frac{\mathbf{TN}}{\mathbf{TN + FP}} - \mathbf{1} \qquad \text{(Eq.3)}$$

Detection rate (DR) [49] is the total number of existing vulnerabilities the scanner managed to detect in the web application.

$$\mathbf{Detection\ Rate} = \frac{\mathbf{TPR}}{\mathbf{TPR + FNR}} \qquad \text{(Eq.4)}$$

$$\mathbf{Accuracy\ Rate} = \frac{\mathbf{TP + FNR}}{\mathbf{TPR + FNR + FPR}} \qquad \text{(Eq.5)}$$

Where:

True Positive (TP) [50] is the total number of correctly detected vulnerabilities by a scanner.

False Positive (FP) [51] is the total number of non-existing vulnerabilities detected by the a scanner.

True Negative (TN) [51] is where there is no existence of vulnerability in the web application.

False Negative (FN) [50] is the tool inability to detect vulnerability in the web application.

5. EXPERIMENTAL RESULTS

This section of the study presents experimental results and evaluation of the tools effectiveness based on the afforementioned metrics.

5.1 Comparison of the scanners detection rate

We evaluated the detection rate of the tools in DVWA and WebGoat based on their numerical measure (i.e. the total number of existing vulnerabilities the scanner managed to detect) and time efficiency (i.e. the time taken to detect vulnerabilities within the shortest possible time) in the web application. It must be noted that vulnerabilities that were not detected in both DVWA and in WebGoat were not used in the analysis due to the required number of pages stipulated.

5.1.1 Numerical measure

Table 1, presents the detection capabilities of the tools in DVWA for the vulnerabilities under study. Vulnerabilities (i.e. CE, BF MC, XSS, GI, GP, and SQL). We observed that the scanners detected known vulnerabilities such as CE, XSS and SQL in DVWA. There is much variation in the detection capabilities of the individual scanners as they achieve different result. For example, OWASP Zap discovered 1 CE, 19 XSS and 6 SQL. Skipfish detected 3 CE, 1 XSS and 2 SQL. Vega detected the highest number of 12 SQL vulnerability in DVWA and ZAP detected the highest number of 19 XSS vulnerability.

Similarly, based on the experimental results presented in Table 1, we are able to see the detection capabilities of each scanner in identifying vulnerabilities in the two vulnerable web applications under study. The vulnerabilities that were detected by the scanner in WebGoat are XSS, BF, and GP. Although no scanner detected all the vulnerabilities in WebGoat, however, the individual detectability is a clear indication that the tools are developed differently and the depth of each strength and weakness invariably also differs.

Again, vulnerabilities such as DoS, CE, HRS, and were not detected by all the scanners. The reasons for this gap could be in two-fold, either the tools are not capable of detecting such vulnerabilities in WebGoat because of their internal functionality or the said vulnerability may not exist in WebGoat.

5.1.2 Time efficiency

We equally evaluated the time efficiency of each scanner. The processing time for each scanner is calculated in seconds by the start time minus the completion time. We recorded the elapsed scanning time for each scanner in both DVWA and Web Goat.

The running time for DVWA ranges from 60 seconds to 180 seconds,360 seconds and 2400 seconds and WebGoat ranges from 60 seconds to 120 seconds and 900 seconds. We also observed that, the running time for vulnerability detection in web application differ from one application to other. For instance, the running time for ZAP to detect vulnerability in DVWA and WebGoat is 360 seconds and 60 seconds respectively. The time differences by the scanners could be attributed to the URL injections points of webpages by the scanners. For example, a smaller number of 2 or 4 URL injection point will demand a few seconds (i.e. 60) than an injection point of 9 or 10 (which could be 120 seconds). Again, the scan profile of the tools for vulnerability detection could vary the detection time. More so, the variations of the tools detection speed (Time) could be attributed to the internal component of the application.

5.2 Results Evaluation

We present a detailed evaluation of the tools based on the standard evaluation metrics discussed in section IV-E.

5.2.1 Detection and accuracy rate analysis of scanners

In this section we evaluate the performance of the scanners base on their detection and accuracy rate in DVWA and in WebGoat. The obtained experimental results are presented in Table 2.

TABLE 2: COMPARISON OF THE SCANNERS DETECTION AND ACCURACY RATE IN DVWA AND WEBGOAT

scanner	DVWA		WebGoat	
	Detection Rate	Accuracy Rate	Detection Rate	Accuracy Rate
ZAP	100%	54.1%	100%	54%
Skipfish	100%	75%	23.5%	94.4%
Arachni	75%	66.6%	41.1%	80.9%
IronWASP	80%	66.6%	5.8%	100%
Vega	100%	56.2%	58.8%	73.9%

From Table 2, it is evidently clear that all the tools obtained a high detection rate in DVWA ranging from 80% to 100%. However, the detection rate in WebGoat is low for all the tools except ZAP which obtained a detection rate of 100%. This is an indication that the detection capability of open source scanners varies from one application to another. This is because of the different penetration testing approaches implemented by the individual scanners to detect vulnerabilities in web application. It is noticeable from the results presented that, the accuracy rate of the scanners in both DVWA and WebGoat is higher. For example, IronWASP recorded accuracy rate of 100% followed by Skipfish and Arachni with accuracy rate of 94.4% and 80% respectively. Again, we observed that, comparatively the accuracy rate in DVWA is low though the detection rate is high, this is due to significant number of false positives reported by the tools. From the experimental results, we observed that no single scanner detected vulnerability of all types in web applications. This is because of the uniqueness of the scanners hence, the number of variations in the detection and accuracy rate.

5.2.2 *Precision and recall analysis of scanners*

In this study both precision and recall metrics are measured in the range of 0-100%. For instance, an effective tool whose detection has no false negative and false positive would have a value of 100% for precision and recall. Table 3, presents the precision and recall values of scanners for both DVWA and WebGoat.

TABLE 3: PRECISON AND RECALL COMPARISON OF SCANNERS

scanner	DVWA		WebGoat	
	Precision	Recall	Precision	Recall
ZAP	54.1%	100%	54%	100%
Skipfish	75%	75%	80%	23.5%
Arachni	60%	75%	63%	70%
IronWASP	80%	50%	100%	6.2%
Vega	56.2%	100%	62.5%	58.8%

In this study both precision and recall metrics are measured in the range of 0-100%. For instance, an effective tool whose detection has no false negative and false positive would have a value of 100% for precision and recall. Table 3, presents the precision and recall values of the scanners for both DVWA and WebGoat. From the Table 3, ZAP and Vega obtained a recall value of 100% in DVWA. These recall values are relatively high which is an indication of the tools ability to detect rear vulnerabilities. Similarly, there are variations in the tools precision values which could be attributed to the tool's uniqueness in vulnerability detection.

For example, IronWASP and Skipfish obtained a precision value of 80% and 75% respectively in DVWA. ZAP and arachni obtained a precision value of 54% and 62.5% respectively in WebGoat which is an indication that the tools detected vulnerabilities that are actually not correctly classified as rear vulnerability (false positive). Skipfish and IronWASP obtained very low recall result in WebGoat 23.5% and 6.2% respectively. This means the inability of the aforementioned tools to detect known vulnerabilities in WebGoat. Details of the precision and recall values for each scanner is presented in Table 3.

5.2.3 *Youden index.*

This section reports the empirical analysis regarding the Youden index of the scanners. As explained in section 4-E, the Youden index evaluates the performance of a tool's diagnostic tests with values ranging from -1 to 1.

TABLE 4: COMPARISON OF THE SCANNERS YOUDEN INDEX IN DVWA

Scanner	DVWA				
	TP	TN	FN	FP	Yi
ZAP	26	3	0	22	0.12
Skipfish	6	3	0	2	0.6
Arachni	6	3	2	4	0.17
Vega	18	3	0	14	0.17
IronWASP	4	3	1	1	0.55

From Table 4, it can be Skipfish obtained the highest Youden index of 0.6 which implies the effectiveness of the scanner in detecting known vulnerabilities in web application with little or no false positive. This is followed by IronWASP with a Youden index of 0.55. Arachni and Vega both obtained a Youden of 0.17 respectively. ZAP obtained the lowest Youden index of 0.12 in DVWA.

TABLE 5: COMPARISON OF THE SCANNERS YOUDEN INDEX IN WEBGOAT

Scanner	WEBGOAT				
	TP	TN	FN	FP	Yi
ZAP	27	3	0	23	0.11
Skipfish	4	3	0	1	0.75
Arachni	7	3	10	4	0.15
Vega	10	3	7	6	0.10
IronWASP	1	5	16	0	0.05

Skipfish outperformed all the scanners by obtaining a Youden index of 0.6 and 0.75 in DVWA and in WebGoat respectively. This is an indication of the tool superiority in detecting vulnerabilities in web application among the other open source scanners. The imbalance variations of the tool Youden index is an indication that a number of open-source scanners can function effectively in detecting security vulnerabilities in web application. Thus, licensing alone should not be used as a standard metric for measuring the effectiveness of a tool. Hence, the aforementioned open-source scanners can be used by security experts for vulnerability detection.

5.2.4 *Lessons learned*

Open-source WAVS are mostly used by vendor for testing web application; however, this study and other existing research have proven the variation in their performance in vulnerability detection. Therefore, in our quest to examine the detection capabilities of open-source scanner based on the aforementioned standard evaluation metrics, we made the following interesting observations:

- The difficulty of open-source WAVS to detect a couple of vulnerabilities in web application is due to the location of the said vulnerability which is preceded by a similar exploited one which makes it difficult to detect the former.

- Although, open source scanners such as ZAP, Skipfish, Vega, and Arachni have high detection rate, they also obtained significant number of false positive which lowers the accuracy rate of the tools.
- We anticipated that, the consequences of high false positives rate in most open source WAVS can render developers to spend scarce resource trying to find solution to vulnerabilities that actually do not exist in web application.
- The report generated by the scanners should not be in a format difficult for users to interpret and understand (e.g. HTML and XML). We rather recommend a user-friendly format such as PDF and Word.
- We observed diversification of the scan result; hence we recommend an approach that can integrate the scanners to compliment the weakness and strength of each other in detecting vulnerabilities.
- We observe that the open source scanners are fairly effective for detecting known vulnerabilities namely Command Execution, Cross-Site Scripting, and SQL injection.
- The functionality of open source scanners should be improved to detect known and unknown web application vulnerability.
- Developers of commercial and open source scanner are always in a hurry to build on customized tool to meet customer requirement as a result most applications and tools come out with a lot of vulnerabilities. Therefore, the development of scanners should be standardized to sanitize the system.

6. THREAT TO VALIDITY

In this section, we discuss the internal and external threat to validity. Threat to internal validity relate to the total number of vulnerabilities in our experimental vulnerable web applications, damn vulnerable web application and WebGoat. However, we estimated the total number of vulnerabilities by the aggregation of the scanners true positive that form a true representation for our experiment. More so, we had challenges in configuring these tools. This is because their functionalities were not compatible with the Java platform (new version) we were using.

We need to try several versions which may be limited in function to carry out the experiment. Threat to external validity relate to the generalization of our results. In this study, we used vulnerability data in two vulnerable web applications to verify the efficiency of the tools. In the future, we will reduce this threat by exploring other vulnerabilities and another implementation tool.

7. CONCLUSION AND FUTURE DIRECTION

In this paper we assessed and analyzed five open-source WAVS, namely OWASP ZAP, Skipfish, Arachni, Iron WASP and Vega using a publicly available web project called Damn Vulnerable Web Application (DVWA) and WebGoat. Additionally, we evaluated the performance of these web application vulnerability scanners using the OWASP benchmark metrics to determine the scanners precision, recall and Youden index, so we can make conclusions with regards to the scanners performance and effectiveness in detecting vulnerabilities. Our findings show that, open source web application vulnerability scanners are very effective in detecting vulnerabilities in web applications. For example, OWASP ZAP and Skipfish are superior in detecting common vulnerabilities such as command execution, cross-site scripting, and SQL injection vulnerabilities. ZAP, Skipfish and Vega obtained a detection rate of 100% and accuracy rate of 54.1%, 75% and 66.6% in DVWA. The detection rate of the tools in WebGoat was not encouraging, Skipfish and Arachni obtained 23.5% and 41.4% respectively. Skipfish obtained the best accuracy rate of 75% in DVWA and 94.4% in WebGoat.

Similarly, Arachni and IronWASP obtained 89.9% and 100% accuracy rate in WebGoat. The results are indication of the tools uniqueness in detecting vulnerabilities in web applications. Furthermore, Skipfish obtained the highest Youden index of 0.7 and 0.6 in DVWA and WebGoat, which makes the scanner superior than the all the studied tools.

8. REFERENCES

[1] D. Sagar, S. Kukreja, J. Brahma, S. Tyagi, and P. Jain, "Studying open source vulnerability scanners for vulnerabilities in web applications," *Institute of Integrative Omics and Applied Biotechnology Journal,* vol. 9, pp. 43-49, 2018.

[2] P. Baral, "Web application scanners: a review of related articles [Essay]," *IEEE Potentials,* vol. 30, pp. 10-14, 2011.

[3] N. Antunes and M. Vieira, "Benchmarking vulnerability detection tools for web services," in *Proceedings of the 2010 IEEE International Conference on Web Services (ICWS),* 2010, pp. 203-210.

[4] M. Vieira, N. Antunes, and H. Madeira, "Using web security scanners to detect vulnerabilities in web services," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks* 2009, pp. 566-571.

[5] M. Zalewski, N. Heinen, and S. Roschke, "Skipfish-web application security scanner," ed: URL: http://code. google. com/p/skipfish/(visited on 06/03/2012), 2011.

[6] I. M. Babincev and D. V. Vuletić, "Web application security analysis using the kali linux operating system," *Vojnotehnički glasnik,* vol. 64, pp. 513-531, 2016.

[7] N. Suteva, D. Zlatkovski, and A. Mileva, "Evaluation and testing of several free/open source web vulnerability scanners," *Proceedings of the 10th Conference for Informatics and Information Technology (CIIT 2013),* 2013.

[8] Y.-H. Tung, S.-S. Tseng, J.-F. Shih, and H.-L. Shan, "A cost-effective approach to evaluating security vulnerability scanner," in *Proceedings of the 15th Asia-Pacific Symposium on Network Operations and Management (APNOMS), 2013* 2013, pp. 1-3.

[9] N. Antunes and M. Vieira, "Detecting SQL injection vulnerabilities in web services," in *Proceedings of the Fourth Symposium on Dependable Computing* 2009, pp. 17-24.

[10] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," in *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS),*

Proceedings of 2015 IEEE 8th International Conference on 2015, 2015, pp. 399-402.

[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 *Proceedings of the 10th International Conference on Internet Technology and Secured Transactions (ICITST)*, 2015, pp. 186-191.

[12] J. Fonseca, M. Vieira, and H. Madeira, "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks," in *Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, 2007, pp. 365-372.

[13] J. Fonseca, M. Vieira, and H. Madeira, "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks," in *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, 2007, pp. 365-372.

[14] L. Suto, "Analyzing the accuracy and time costs of web application security scanners," *San Francisco, February,* 2010.

[15] J. Fonseca, M. Vieira, and H. Madeira, "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks," in *Proceedings of 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, 2007, pp. 365-372.

[16] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," in *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015 IEEE 8th International Conference on*, 2015, pp. 399-402.

[17] N. I. Daud, K. A. A. Bakar, and M. S. M. Hasan, "A case study on web application vulnerability scanning tools," in *2014 Science and Information Conference*, 2014, pp. 595-600.

[18] "https://www.owasp.org/index.php/Benchmark."

[19] A. Baratloo, M. Hosseini, A. Negida, and G. El Ashal, "Part 1: simple definition and calculation of accuracy, sensitivity and specificity," 2015.

[20] C. J. Van Rijsbergen, "A non-classical logic for information retrieval," *The computer journal*, vol. 29, pp. 481-485, 1986.

[21] M. Alsaleh, N. Alomar, M. Alshreef, A. Alarifi, and A. Al-Salman, "Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners," *Security and Communication Networks*, vol. 2017, 2017.

[22] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic, "Secubat: a web vulnerability scanner," in *Proceedings of the 15th International Conference on World Wide Web*, 2006, pp. 247-256.

[23] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*: Prentice Hall PTR, 2002.

[24] M. Sutton, A. Greene, and P. Amini, *Fuzzing: brute force vulnerability discovery*: Pearson Education, 2007.

[25] E. F. R. G. V. Okun, P. E. Black, and E. Dalci, "Building a Test Suite for Web Application Scanners."

[26] O. Hamed and N. Kafri, "Performance Prediction of Web Based Application Architectures Case Study: .NET vs. Java EE," *International Journal of Web Applications*, vol. 1, 2009.

[27] J. C. Fonseca, M. Vieira, and H. Madeira, "Correlating security vulnerabilities with software faults," 2007.

[28] H. Le and P. Loh, "Unified approach to vulnerability analysis of web applications," in *AIP Conference Proceedings*, 2008, pp. 155-159.

[29] P. E. Black and E. Fong, "Proceedings of Defining the State of the Art in Software Security Tools Workshop," *NIST Special Publication*, vol. 500, p. 264, 2005.

[30] S. Panguluri, W. Phillips, and P. Ellis, "Cyber security: protecting water and wastewater infrastructure," in *Handbook of water and wastewater systems protection*, ed: Springer, 2011, pp. 285-318.

[31] S. Zhang, D. Caragea, and X. Ou, "An empirical study on using the national vulnerability database to predict software vulnerabilities," in *Proceedings of the International Conference on Database and Expert Systems Applications*, 2011, pp. 217-231.

[32] M. Abedin, S. Nessa, E. Al-Shaer, and L. Khan, "Vulnerability analysis for evaluating quality of protection of security policies," in *Proceedings of the 2nd ACM Workshop on Quality of Protection*, 2006, pp. 49-52.

[33] J. A. Wang and M. Guo, "Vulnerability categorization using Bayesian networks," in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, 2010, p. 29.

[34] P. Anbalagan and M. Vouk, "On mining data across software repositories," in *Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories, 2009. MSR'09.* , 2009, pp. 171-174.

[35] S. Evans, "Securing WebGoat using ModSecurity, summer of code 2008," *OWASP beta level, OWASP Foundation,* 2008.

[36] R. Mohammed, "Assessment of Web Scanner Tools," *International Journal of Computer Applications (0975-8887),* vol. 133, 2016.

[37] K. McQuade, "Open source web vulnerability scanners: the cost effective choice," in *Proceedings of the Conference for Information Systems Applied Research ISSN*, 2014, p. 1508.

[38] N. Teodoro and C. Serrão, "Automating Web Applications Security Assessments through Scanners," *Web Application Security*, p. 48.

[39] N. Suteva, D. Zlatkovski, and A. Mileva, "Evaluation and testing of several free/open source web vulnerability scanners," 2013.

[40] " OWASP. OWASP Benchmark. Available: https://www.owasp.org/index.php/Benchmark, 2017.," pp. 1-6.

[41] B. Mburano and W. Si, "Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark," in *Proceedings of the 26th International Conference on Systems Engineering (ICSEng)*, 2018, pp. 1-6.

[42] E. n. İ. Tatli and B. r. Urgun, "WIVET—benchmarking coverage qualities of web crawlers," *The Computer Journal,* vol. 60, pp. 555-572, 2017.

[43] N. A. Aziz, S. N. Z. Shamsuddin, and N. A. Hassan, "Inculcating Secure Coding for beginners," in *Proceedings of the International Conference on Informatics and Computing (ICIC), ,* 2016, pp. 164-168.

[44] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," in *Proceedings of the 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015*, 2015, pp. 399-402.

[45] M. El, E. McMahon, S. Samtani, M. Patton, and H. Chen,

"Benchmarking vulnerability scanners: An experiment on SCADA devices and scientific instruments," in *Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2017, pp. 83-88.

[46] S. Idrissi, N. Berbiche, F. Guerouate, and M. Shibi, "Performance evaluation of web application security scanners for prevention and protection against vulnerabilities," *International Journal of Applied Engineering Research*, vol. 12, pp. 11068-11076, 2017.

[47] Y.-H. Tung, S.-S. Tseng, J.-F. Shih, and H.-L. Shan, "W-VST: A Testbed for Evaluating Web Vulnerability Scanner," in *Proceeding of the 14th International Conference on Quality Software*, 2014, pp. 228-233

[48] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, pp. 32-35, 1950.

[49] H. Holm, T. Sommestad, J. Almroth, and M. Persson, "A quantitative evaluation of vulnerability scanning," *Information Management & Computer Security*, vol. 19, pp. 231-247, 2011

[50] J. Akosa, "Predictive accuracy: a misleading performance measure for highly imbalanced data," in *Proceedings of the SAS Global Forum*, 2017, pp. 2-5.

[51] N. Antunes and M. Vieira, "On the metrics for benchmarking vulnerability detection tools," in *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2015, pp. 505-516.